

Les systèmes multi-agents minimaux

Francis van Aeken

▶ **To cite this version:**

Francis van Aeken. Les systèmes multi-agents minimaux. Autre [cs.OH]. Institut National Polytechnique de Grenoble - INPG, 1999. Français. NNT : . tel-00004860

HAL Id: tel-00004860
https://theses.hal.science/tel-00004860

Submitted on 18 Feb 2004

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

présentée par

FRANCIS VAN AEKEN

pour obtenir le grade de
Docteur de l'Institut National Polytechnique de Grenoble
(Arrête ministériel du 30 mars 1992)

Spécialité : Informatique

LES SYSTÈMES MULTI-AGENTS MINIMAUX

Un Modèle Adapté à l'Etude de la Dynamique Organisationnelle dans les Systèmes Multi-Agents Ouverts

Soutenue le 30 mars 1999 devant le jury composé de

M. PHILIPPE JORRAND,	PRESIDENT DU JURY
M. YVES DEMAZEAU,	DIRECTEUR DE THESE
M. JACQUES FERBER,	RAPPORTEUR
M. HENRI LIEBERMAN,	RAPPORTEUR
M. THIERRY BOURON,	EXAMINATEUR
M. ANDRZEJ DUDA,	EXAMINATEUR

Thèse préparée au sein du Laboratoire LEIBNIZ

à la mémoire d'Alan Turing, 1912-1954

THESE — RESUME

La thèse soutenue est la suivante :

— Un modèle minimal des Systèmes Multi-Agents peut être fondé sur le concept du **couple**, et le principe « **qui se ressemble, s'assemble** ».

— Un tel modèle peut mener à une meilleure compréhension de la dynamique organisationnelle dans les Systèmes Multi-Agents.

— Un tel modèle peut être appliqué à des problèmes importants.

Résumé :

En approchant la fin du Millénaire, nous sommes témoins de la genèse d'un nouveau monde : l'univers électronique composé des millions d'ordinateurs interconnectés par l'Internet.

Dans ce monde, des processus informatiques autonomes migreront d'ordinateur à ordinateur. En migrant, ces entités, couramment appelées « agents », exploiteront mieux les ressources disponibles et assureront mieux leur survie. De plus, pour des raisons d'efficacité, ils s'organiseront en sociétés.

Les sociétés d'agents artificiels sont étudiées dans le domaine des Systèmes Multi-Agents (SMA). La notion d'organisation est essentielle dans le domaine. Dans le cadre de notre thèse, nous proposons les Systèmes Multi-Agents Minimaux (SMAM), un modèle permettant l'étude de l'organisation des agents à un niveau fondamental.

Le modèle est basé sur la notion du couple : les agents sont récursivement organisés en couples. Le comportement des agents est guidé par le principe « qui se ressemble, s'assemble ». En utilisant ce modèle, nous pouvons quantitativement analyser un nombre de phénomènes liés à la dynamique organisationnelle des SMA.

De plus, nous pouvons directement appliquer notre modèle dans le contexte de l'Internet. Nous avons développé l'application « FRIENDS », dans laquelle les utilisateurs sont représentés par des agents s'organisant comme un SMAM. Elle permet aux utilisateurs de retrouver des groupes et des individus partageant des intérêts communs.

Le modèle est nouveau et beaucoup de travail reste à faire. Toutefois, à cause de sa nature universelle et ses applications potentielles sur l'Internet, nous sommes confiants en son avenir.

Mots-clés :

Systèmes Multi-Agents, dynamique organisationnelle, Internet

THESIS — ABSTRACT

The supported thesis is the following:

— A minimal model of Multi-Agent Systems can be based on the concept of the **couple**, and the principle "**likes seek likes**".

— Such a model can lead to a better understanding of the organisational dynamics in Multi-Agent Systems.

— Such a model can be applied to important problems.

Abstract:

While approaching the end of the Millennium, we are witnessing the genesis of a new world: the electronic universe composed of the millions of computers connected by the Internet.

In this world, autonomous computer processes will migrate from computer to computer. By migrating, these entities, typically called "agents", will better exploit the available resources and secure their survival. Moreover, for reasons of efficiency, they will organise themselves into societies of agents.

Societies of artificial agents are studied within the domain of Multi-Agent Systems (MAS). The notion of organisation is essential in the domain. Within the context of our thesis, we propose the model of Minimal Multi-Agent Systems (MMAS), which allows for the study of the organisation of agents at a fundamental level.

The model is based on the notion of the couple: agents are recursively organised in couples. The behaviour of the agents is guided by the principle "likes seek likes". Using this model, we can quantitatively analyse a number of phenomenon related to the organisational dynamics of MAS.

Moreover, we can directly apply our model within the context of the Internet. We have developed the application "FRIENDS", in which users are represented by agents that organise themselves as a MMAS. The application allows users to find groups and individuals sharing common interests.

The model is new and a lot of work is still to be done. Still, because of its universal nature and its potential applications of the Internet, we are confident in its future.

Keywords:

Multi-Agent Systems, organisational dynamics, Internet

THESIS — SAMENVATTING

De verdedigde thesis luidt als volgt:

— Een minimaal model van Multi-Agent Systemen kan worden gebaseerd op de notie van het **koppel**, en het principe "**soort zoekt soort**".

— Zulk een model kan leiden tot een beter inzicht in de organisatorische dynamiek in Multi-Agent Systemen.

— Zulk een model kan verder worden toegepast op belangrijke problemen.

Samenvatting:

Op de vooravond van het nieuwe Millenium zijn we getuige van de geboorte van een nieuwe wereld: het elektronische universum bestaande uit de miljoenen computers verbonden door het Internet.

In deze wereld zullen autonome computerprocessen van computer tot computer migreren. Al migrerend zullen deze entiteiten, "agents" genaamd, de beschikbare middelen optimaal uitbuiten en hun kansen op overleven vergroten. Om alles zo efficient mogelijk te laten verlopen zullen ze zich organiseren in sociale groepen.

Samenlevingen van kunstmatige agents worden bestudeerd in het kader van Multi-Agent Systemen (MAS). De notie van organisatie is essentieel in deze discipline. In de context van onze thesis stellen wij Minimale MAS voor (MMAS), een model dat zich leent tot de fundamentele studie van de organisatie van agents.

Ons model is gebaseerd op het concept van het koppel: agents zijn recursief georganiseerd in koppels. Het gedrag van de agents wordt bepaald door het principe "soort zoekt soort". Met behulp van dit model kunnen we kwantitatief een aantal phenomenen bestuderen rond de organisatorische dynamiek van MAS.

Bovendien kunnen we ons model direct toepassen binnen de context van het Internet. We hebben de toepassing "FRIENDS" ontwikkeld, waarin gebruikers worden vertegenwoordigd door agents die zich organiseren als een MMAS. Deze gebruikers kunnen dan gemakkelijk groepen of individuen vinden die dezelfde interesses delen.

Ons model is nieuw en er is nog veel werk voor de boeg. Omwille van zijn universele natuur en zijn potentiële toepassingen op het Internet echter, hebben we volop vertrouwen in zijn toekomst.

Sleutelwoorden:

Multi-Agent Systemen, organisatorische dynamiek, Internet

REMERCIEMENTS

Ce manuscrit présente des recherches que je n'aurai jamais pu réaliser sans le support, les conseils et l'amitié de nombreuses personnes.

Tout d'abord je remercie M. Yves Demazeau, Chargé de Recherches CNRS et responsable de l'équipe MAGMA - LEIBNIZ, qui m'a accueilli dans son équipe et qui a dirigé ma thèse. C'est la deuxième fois que nous travaillons ensemble et cette fois encore notre collaboration a porté ses fruits.

Je remercie également M. Philippe Jorrand, Directeur de Recherches CNRS et directeur du laboratoire LEIBNIZ - IMAG, de m'avoir offert cet excellent environnement de travail et de présider le Jury de Thèse.

J'exprime ma reconnaissance sincère aux autres membres du Jury :

M. Thierry Bouron, chercheur au Centre de Recherche et de Développement de France Télécom,
M. Andrzej Duda, Professeur à l'Institut National Polytechnique de Grenoble,
M. Jacques Ferber, Professeur à l'Université Montpellier II, et
M. Henri Lieberman, Professeur au Massachusetts Institute of Technology.

Je remercie France Télécom et la Communauté Européenne pour leur soutien financier, sous la forme du contrat CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -Leibniz, et de la bourse « Fixed Contribution Contract for Training through Research » N° ERBFMBICT961525.

Un grand merci à Daniel Genovese qui, pendant toute une année, a contribué à cette thèse en implémentant FRIENDS Online. Merci aussi à Frédéric Maffray et à Myriam Preissman d'avoir démontré la NP-difficulté de l'organisation optimale d'un système FRIENDS.

Je remercie chaleureusement tous ceux qui ont relu des parties de ce manuscrit, en particulier Guillaume Chicoisne, Marc-Philippe Huget, Jean-Luc Koning, Astrid Mussi, Sylvie Pesty Pierre-Michel Ricordel et Koen Van Aeken. Evidemment, je suis le seul responsable des erreurs éventuelles restantes.

Je remercie tous les membres et anciens membres de l'équipe MAGMA qui m'ont appris tellement de choses. En plus de ceux déjà mentionnés, je pense particulièrement à Christof Baeijs, Nils Ferrand, Pierre Ferrant, Jarek Kozlak, Michel Occello, Alexandre Ribeiro et Vera Strube De Lima.

Je n'oublie pas Michel Plu de France Télécom, avec lequel les discussions ont été particulièrement fructueuses.

Je dois tant de choses à mon ami Tom et à ma copine Astrid, qui m'ont inconditionnellement supporté et qui me sont tellement chers. Merci, les gars !

Enfin, je remercie tout spécialement ma mère, mon père, mes frères et ma sœur. Leur amour et leur foi m'ont accompagné tout au long de cette recherche.

PLAN

Ce mémoire compte cinq parties. Dans la première nous ébauchons le contexte dans lequel nous avons développé notre modèle. Le modèle en lui-même est introduit dans la seconde. Dans la troisième partie, nous faisons la transition entre la théorie présentée dans la partie II et les applications détaillées dans la partie IV. Nous concluons le mémoire dans la cinquième partie.

I.1 La première partie constituant l'introduction du mémoire, compte un seul chapitre. Dans ce chapitre, nous présentons le contexte dans lequel nos travaux doivent être interprétés.

II.1 La deuxième partie est composée de cinq chapitres. Dans le premier nous présentons les motivations sur lesquelles s'appuient nos travaux.

II.2 Dans le deuxième chapitre, nous définissons les Systèmes Multi-Agents Minimaux (SMAM), en spécifiant leur structure et leur comportement au niveau macroscopique et microscopique.

II.3 Le troisième chapitre détaille les algorithmes implémentant les SMAM. Nous les présentons et les comparons. Nous présentons également un *benchmark* pour des agents cognitifs.

II.4 Nous présentons quelques réflexions sur notre modèle dans le quatrième chapitre. Nous y discutons également de la visualisation des SMAM et de leur relation aux arbres binaires.

II.5 Dans le cinquième chapitre, nous situons notre modèle dans le domaine des Systèmes Multi-Agents, la Systémique et la Vie Artificielle.

III.1 La troisième partie comporte trois chapitres. Dans le premier nous discutons de la validation de modèles scientifiques. Nous y introduisons également les SMAM augmentés et le concept FRIENDS**.**

III.2 Dans le deuxième chapitre nous présentons FRIENDS Offline, notre plate-forme d'expérimentation.

III.3 Le troisième chapitre est dédié aux agents mobiles. Nous y discutons de leur fonctionnalité et de leur implémentation. Nous y présentons également les principaux systèmes existants.

IV.1 Deux chapitres constituent la quatrième partie. Le premier présente FRIENDS Numbercruncher. Nous situons cette application dans le contexte du regroupement hiérarchique et l'évaluons.

IV.2 Dans le deuxième chapitre nous présentons FRIENDS Online. Nous le situons par rapport aux techniques classiques et au *Community Ware*. Nous présentons également l'évaluation du système.

V.1 Dans la dernière partie, composée d'un seul chapitre, nous présentons l'évaluation globale et les perspectives de notre modèle et de nos applications.

TABLE DES MATIERES

Thèse — Résumé 5
Thesis — Abstract 7
Thesis — Samenvatting 9
Remerciements 11
Plan 13
Table des Matières 15
Liste des Figures, Graphiques et Tableaux 21
Conventions 25

PARTIE I INTRODUCTION 27

I.1 LE CONTEXTE 29
I.1.1 Introduction 29
I.1.2 L'Informatique 29
I.1.3 L'Intelligence Artificielle 32
I.1.4 Les Agents Autonomes 33
La spécialisation de l'IA 34
La renaissance de l'IA 34
Autonomie 35
Le monde physique 36
Intelligence sans raisonnement, sans représentation 36
Les architectures réactives 37
L'architecture « subsumption » 38
La cybernétique 38
L'invasion des agents 39
La définition de la notion d'agent 40
I.1.5 Les Systèmes Multi-Agents 41
I.1.6 La dynamique organisationnelle des Systèmes Multi-Agents 43
I.1.7 Le maintien de l'intégrité fonctionnelle 45
I.1.8 L'Internet 47
Le World Wide Web 47
Deux mondes 48

PARTIE II LE MODELE 51

II.1 MOTIVATIONS 53

II.1.1 Un modèle minimal 53

II.1.2 Un modèle systémique 55

II.1.3 Un modèle sans connotations 56

II.2 DEFINITION DES SYSTEMES MULTI-AGENTS MINIMAUX 59

II.2.1 Agents atomiques et agents composés 59

Une définition récursive 59

Nombre de SMAM différents 62

Structure et organisation 63

II.2.2 Modéliser en base deux 64

La conférence 64

Quelques organisations 66

II.2.3 Nommer les agents 70

II.2.4 Mesurer les SMAM 72

Taille et taille réelle 73

Equilibre 74

Entropie 75

Calculer l'entropie 78

La distance entre SMAM comme mesure de différence 80

La distance entre SMAM comme mesure d'éloignement 80

II.2.5 L'évolution naturelle des SMAM 82

Evolution de la taille 83

Evolution de l'équilibre 84

II.2.6 Le comportement des agents 86

II.2.7 Les définitions essentielles des SMAM 87

II.3 ALGORITHMES CENTRALISES ET DISTRIBUES 89

II.3.1 L'algorithme et sa classification 89

Concept, algorithmes et implémentations des SMAM 89

Critères de classification 90

II.3.2 Présentation et évaluation des algorithmes 91

L'algorithme « global aggregated » 93

L'algorithme « local aggregated » 96

L'algorithme « global E-matcher II » 99

L'algorithme « local E-matcher » 101

Comparaison des algorithmes 103

II.3.3 Un *benchmark* pour les agents cognitifs 103
Un problème pertinent 104
Définition du benchmark 104
II.3.4 Les SMAM hétérogènes 106

II.4 REFLEXIONS SUR LE MODELE 109
II.4.1 Evolution du modèle 109
II.4.2 Le couple : ensemble ou séquence ? 110
II.4.3 Attraction ou répulsion ? 112
II.4.4 « Les extrêmes s'attirent » ? 114
II.4.5 Visualiser les SMAM 116
Critères de classification 116
Les représentations actuelles 117
La représentation originale 117
La représentation lexicale 119
La représentation arborescente 119
La représentation triangulaire 123
La représentation « arbre H » 125
Comparaison des représentations 127
Le côté esthétique 127
II.4.6 Les SMAM et les arbres binaires 128
Définition des arbres binaires 128
Les arbres binaires complets 129
Les arbres binaires complets et non ordonnés 129
Les arbres binaires de recherche 129

II.5 SITUER LE MODELE 131
II.5.1 Les SMAM et les Systèmes Multi-Agents 131
Les agents 131
L'environnement 132
Les interactions 134
L'organisation 135
Migration 139
II.5.2 Les SMAM et la Systémique 142
La théorie de l'information et du codage 142
La thermodynamique 144
II.5.3 Les SMAM et la Vie Artificielle 145
Autopoièse 145
Maximisation de la taille des SMAM 149
La Thermodynamique Artificielle 152

PARTIE III DU MODELE VERS LES APPLICATIONS 155

III.1 APPLIQUER LES SYSTEMES MULTI-AGENTS MINIMAUX 157

III.1.1 La validation de modèles scientifiques 157

III.1.2 Les SMAM augmentés 157

III.1.3 FRIENDS 161

III.2 FRIENDS OFFLINE, OUTIL D'EXPERIMENTATION 163

III.2.1 Une plate-forme interactive 163

L'essentiel de la plate-forme 164

Un prototype de FRIENDS 166

Entropie et harmonie 168

L'algorithme « global attributed III » 170

III.2.2 L'implémentation de FRIENDS Offline 171

Exploiter la récursivité 171

III.3 LES AGENTS MOBILES 175

III.3.1 La fonctionnalité des agents mobiles 175

La mobilité 175

Exploiter la bande passante 178

L'importance de la mobilité 180

La mobilité et la migration 182

III.3.2 Implémenter la mobilité 182

Les mécanismes de base 182

Le « classloader » en Java 185

Le protocole de migration 187

III.3.3 Les systèmes existants 187

Agent Tcl (D'Agents) 187

Aglets 188

Concordia 188

Mole 188

Odyssey 188

Telescript 188

Voyager 189

Systèmes moins connus 189

Une courte évaluation 189

Les normes et les agents mobiles 190

III.3.4 La technologie des composants 191

PARTIE IV LES APPLICATIONS 193

IV.1 LE REGROUPEMENT HIERARCHIQUE 195

IV.1.1 Introduction au regroupement hiérarchique 195

La mesure de similarité 196

Le regroupement hiérarchique 197

IV.1.2 Les SMAM augmentés et le regroupement hiérarchique 200

La mesure de distance de FRIENDS 201

Un problème NP-difficile 203

IV.1.3 FRIENDS Numbercruncher 203

Première phase : « Start Analysis » 204

Deuxième phase : « Build Dictionary » 205

Troisième phase : « Create Population » 205

Quatrième phase : « Start Organisation » 205

Cinquième phase : « Generate Report » 206

Performance 207

IV.1.4 Evaluation 208

QuiQuoiOù 208

Les essais 208

IV.2 LA RECHERCHE ET LA CLASSIFICATION D'INFORMATIONS SUR LE WEB 211

IV.2.1 Trouver des informations sur le Web 211

Les robots 211

La classification semi-automatique 213

Les méta-moteurs 214

IV.2.2 *Community Ware* 214

Les travaux à la Kyoto University et à NTT 215

Les travaux au MIT Media Lab 216

Yenta 217

Un Web auto-organisant 220

IV.2.3 FRIENDS Online 220

Principe 220

FRIENDS Online et Yenta 222

Implémentation 224

IV.2.4 Evaluation 227

IV.2.5 FRIENDS Online MAI 228

Le protocole de migration 228

PARTIE V CONCLUSION 233

V.1 EVALUATION ET PERSPECTIVES 235

V.1.1 A propos du modèle 235

Evaluation 235

Perspectives 236

V.1.2 A propos des applications 237

Evaluation 237

Perspectives 238

V.1.3 A propos de la thèse 239

ANNEXES 241

Annexe I — Un Problème NP-Difficile 243

Annexe II — Classification de Services 245

Annexe III — FRIENDS Offline 249

Annexe IV — FRIENDS Numbercruncher 259

Annexe V — FRIENDS Online 267

Annexe VI — Les Systèmes Multi-Agents Minimaux, le CD 275

BIBLIOGRAPHIE 277

Bibliographie 279

LISTE DES FIGURES, GRAPHIQUES ET TABLEAUX

Chapitre I.1

Figure 1 : Le rôle du modèle informatique dans l'Intelligence Artificielle. 33
Figure 2 : La décomposition horizontale et verticale. 38
Figure 3 : Les niveaux différents d'organisation. 44

Graphique 1 : Evolution du nombre de transistors dans les processeurs d'Intel. 31
Graphique 2 : Evolution du nombre d'hôtes sur l'Internet. 47

Tableau 1 : Evolution du nombre de transistors dans les processeurs d'Intel. 30
Tableau 2 : Comparaison de terminologie. 44

Chapitre II.1

Figure 1 : Modèle défini au niveau des différentes disciplines. 57
Figure 2 : Modèle défini à un niveau « sous-disciplinaire ». 57

Chapitre II.2

Figure 1 : L'agent atomique. 60
Figure 2 : L'agent composé. 60
Figure 3 : Représentation triangulaire des SMAM des figures 1 et 2. 61
Figure 4 : SMAM composé de trois agents atomiques. 61
Figure 5 : SMAM composé de quatre agents atomiques (possibilité 1). 62
Figure 6 : SMAM composé de quatre agents atomiques (possibilité 2). 62
Figure 7 : La conférence (temps *t*). 65
Figure 8 : La conférence (temps *t'*). 65
Figure 9a, 9b : La conférence (représentation triangulaire). 66
Figure 10 : Hiérarchie. 66
Figure 11 : SMAM correspondant à l'organisation de la figure 10. 67
Figure 12 : Groupe non structuré. 67
Figure 13 : SMAM correspondant à l'organisation de la figure 12. 67
Figure 14 : Organisation non binaire. 68
Figure 15 : SMAM correspondant à l'organisation de la figure 14. 68
Figure 16 : Organisation hiérarchique non binaire. 68
Figure 17 : SMAM correspondant à l'organisation de la figure 16. 69
Figure 18 : Représentation triangulaire du SMAM de la figure 17. 69
Figure 19 : Nommer les agents. 70
Figure 20 : Nommer les agents (représentation triangulaire). 70
Figure 21 : SMAM simple. 71
Figure 22 : SMAM complexe. 72
Figure 23 : Représentation triangulaire du SMAM de la figure 22. 72
Figure 24 : Arbre binaire comptant une seule feuille. 73
Figure 25 : SMAM très bien équilibré de taille réelle 8. 74
Figure 26 : SMAM très mal équilibré de taille réelle 8. 74
Figure 27 : Valeurs d'entropie des agents d'un SMAM. 75
Figure 28 : Valeurs d'entropie dans les SMAM des figures 25 et 26. 75
Figure 29 : Représentations compactes des SMAM des figures 25 et 26. 78
Figure 30 : Deux SMAM différents. 80
Figure 31 : Valeurs d'entropie des SMAM de la figure 30. 80
Figure 32 : Agent *A* et ses voisins. 81

Figure 33 : Graphe de voisinage du SMAM de la figure 32. 81
Figure 34 : SMAM *S* au temps *t*. 85
Figure 35 : SMAM *S* au temps $t + \Delta t$. 85

Graphique 1 : Nombre de SMAM différents. 63
Graphique 2 : Nombre de SMAM différents (échelle logarithmique). 63
Graphique 3 : Evolution possible de l'entropie du SMAM de la figure 34. 86

Tableau 1 : Nombre de SMAM différents. 63
Tableau 2 : Nommer les agents. 70

Chapitre II.3

Figure 1 : SMAM de taille réelle 32. 92
Figure 2 : Transformation optimale du SMAM de la figure 1. 92
Figure 3 : SMAM de taille réelle 31. 93
Figure 4 : Transformation optimale du SMAM de la figure 3. 93
Figure 5a, 5b : SMAM mal équilibrés. 96
Figure 6 : Oscillation simple. 97
Figure 7 : Oscillation complexe. 98
Figure 8 : A la recherche de l'autre. 102

Graphique 1 : Comportement de l'algorithme « global aggregated » pour $TR(S) = 32$. 94
Graphique 2 : Comportement de l'algorithme « global aggregated » pour $TR(S) = 31$. 95
Graphique 3 : Comportement de l'algorithme « local aggregated » pour $TR(S) = 32$. 97
Graphique 4 : Oscillation simple. 98
Graphique 5 : Oscillation complexe. 98
Graphique 6 : Comportement de l'algorithme « local aggregated » pour $TR(S) = 31$. 99
Graphique 7 : Comportement de l'algorithme « global E-matcher II » pour $TR(S) = 32$. 100
Graphique 8 : Comportement de l'algorithme « global E-matcher II » pour $TR(S) = 31$. 101
Graphique 9 : Comportement de l'algorithme « local E-matcher » pour $TR(S) = 32$. 102
Graphique 10 : Comportement de l'algorithme « local E-matcher » pour $TR(S) = 31$. 102

Tableau 1 : Comparaison des algorithmes. 103
Tableau 2 : Nombre de tests exigés par $B(X, N)$. 106

Chapitre II.4

Figure 1 : Symétrie parfaite. 111
Figure 2 : Correspondance entre particules et agents atomiques. 113
Figure 3 : SMAM « modèle ». 117
Figure 4 : Représentation originale (agent atomique). 117
Figure 5 : Représentation originale (agent composé). 118
Figure 6 : Représentation originale (agent de la figure 3). 118
Figure 7 : Représentation originale, visualisant l'arbre correspondant. 118
Figure 8 : Représentation arborescente (agent atomique). 119
Figure 9 : Représentation arborescente (agent composé). 120
Figure 10 : Représentation arborescente (agent de la figure 3). 120
Figure 11 : Représentation arborescente, visualisant la grille. 121
Figure 12 : Représentation arborescente, visualisant les valeurs d'entropie. 121
Figure 13 : Représentation arborescente, compacte. 121
Figure 14 : Représentation arborescente, compacte, visualisant les valeurs d'entropie. 122
Figure 15 : Représentation arborescente, migration continue. 122
Figure 16 : Triangle à la base de la représentation triangulaire. 123
Figure 17 : Représentation triangulaire (agent atomique). 123
Figure 18 : Représentation triangulaire (agent composé). 124
Figure 19 : Représentation triangulaire (agent de la figure 3). 124
Figure 20 : Représentation triangulaire, bordures larges. 124
Figure 21 : Représentation triangulaire, migration continue. 125
Figure 22 : Représentation « arbre H » (agent atomique). 125

Figure 23 : Représentation « arbre H » (agent composé). 126
Figure 24 : Représentation « arbre H » (agent de la figure 3). 126
Figure 25 : Représentation « arbre H » d'un agent atomique d'ordre 8. 126

Graphique 1 : Distribution de particules. 112
Graphique 2 : Distribution de particules (modèle SMAM). 113
Graphique 3 : Dissonance de deux sons. 115

Tableau 1 : Comparaison des représentations visuelles. 127

Chapitre II.5

Figure 1 : Environnement interne de l'agent *A*. 133
Figure 2 : SMAM correspondant à une hiérarchie extrême. 138
Figure 3 : Intégration de *B* dans *A* (phase initiale). 141
Figure 4 : Intégration de *B* dans *A* (phase intermédiaire). 141
Figure 5 : Intégration de *B* dans *A* (phase finale). 141
Figure 6 : Trois SMAM ayant la signature organisationnelle de la figure 7. 147
Figure 7 : Exemple d'une signature organisationnelle. 147
Figure 8 : Trois SMAM ayant la signature organisationnelle de la figure 9. 147
Figure 9 : Deuxième exemple d'une signature organisationnelle. 147
Figure 10 : SMAM ayant une signature organisationnelle similaire à celle de la figure 7. 148
Figure 11 : SMAM ayant une signature organisationnelle similaire à celle de la figure 9. 148
Figure 12 : Evolution d'un SMAM (phase initiale). 150
Figure 13 : Evolution d'un SMAM (phase intermédiaire). 151
Figure 14 : Evolution d'un SMAM (phase finale). 151

Graphique 1 : Equilibre diminuant d'un SMAM linéaire croissant. 140
Graphique 2 : Immigration et réorganisation. 140
Graphique 3 : Sigmoïde. 149
Graphique 4 : Sinusoïde. 149
Graphique 5 : Croissance de la taille moyenne des agents atomiques d'ordre supérieur. 152

Tableau 1a : Correspondance entre organisations et SMAM (partie 1). 136
Tableau 1b : Correspondance entre organisations et SMAM (partie 2). 137
Tableau 2 : Croissance de la taille moyenne des agents atomiques d'ordre supérieur. 151

Chapitre III.1

Figure 1 : Représentation en base 1 et en base 2 du nombre 7. 158
Figure 2 : SMAM de taille réelle 22 et d'équilibre maximale. 158
Figure 3 : Représentation binaire du nombre 22. 159
Figure 4 : SMAM restreint, représentant plusieurs symboles. 160
Figure 5 : Groupement de symboles dans un SMAM restreint. 160

Tableau 1 : SMAM représentant des symboles externes. 159

Chapitre III.2

Figure 1 : Fenêtre principale de FRIENDS Offline. 164
Figure 2 : Fenêtre « Triangle View » de FRIENDS Offline. 165
Figure 3 : Fenêtre « H-Tree View » de FRIENDS Offline. 165
Figure 4 : Fenêtre « History » de FRIENDS Offline. 166
Figure 5 : Exemple d'une population parfaite. 167
Figure 6 : Organisation optimale de la population de la figure 5. 167
Figure 7 : Organisation optimale alternative de la population de la figure 5. 169
Figure 8 : Population *presque* monotone. 170

Graphique 1 : Relation entre $N(A)$ et $2^{N(A)}$. 169
Graphique 2 : Evolution de l'entropie de trois SMAM augmentés. 171

Chapitre III.3

Figure 1 : Migration (phase 1). 176
Figure 2 : Migration (phase 2). 176
Figure 3 : Migration (phase 3). 176
Figure 4 : Migration (phase 4). 177
Figure 5 : Migration (phase 5). 177
Figure 6 : Interaction à travers le réseau. 177
Figure 7 : Négociation entre agents mobiles. 178
Figure 8 : Négociation entre agents statiques. 178
Figure 9 : Eléments essentiels d'un système « Telescript ». 183
Figure 10 : Implémentation de la mobilité (cas 1). 184
Figure 11 : Implémentation de la mobilité (cas 2). 184
Figure 12 : Implémentation de la mobilité (cas 3). 184
Figure 13 : Implémentation de la mobilité (cas 4). 185

Graphique 1 : Agents statiques vs. agents mobiles (a = 5000). 179
Graphique 2 : Agents statiques vs. agents mobiles (a = 25000). 179

Tableau 1 : Eléments informatiques, statiques et mobiles. 176

Chapitre IV.1

Figure 1 : Exemple de regroupement hiérarchique. 198
Figure 2 : Fenêtre principale de FRIENDS Numbercruncher. 204
Figure 3 : Organisation sous-optimale d'une population. 206
Figure 4 : Organisation optimale de la population de la figure 3. 206

Graphique 1 : Exemple de regroupement. 196
Graphique 2 : Complexité des algorithmes. 200
Graphique 3 : Extrapolation des temps d'organisation. 207

Tableau 1 : Les coefficients de Lance et Williams. 199
Tableau 2 : Comparaison entre la ressemblance et la distance « Manhattan ». 201
Tableau 3 : Mesure de distance déduite de la ressemblance. 202
Tableau 4 : $d(A, C) > d(A, B) + d(B, C)$. 202
Tableau 5 : Temps d'organisation. 207
Tableau 6 : Catégories identifiées par FRIENDS Numbercruncher. 209

Chapitre IV.2

Figure 1 : Structures dans Yenta. 219
Figure 2 : La migration de *D* vers *E* équilibrera le système. 224
Figure 3 : Fenêtre principale de FRIENDS Online « Client ». 225
Figure 4 : Fenêtre principale de FRIENDS Online « Administrator ». 226
Figure 5 : Migration simultanée d'*A* vers *B* et de *C* vers *D*. 229
Figure 6 : Migration simultanée d'*A* vers *D* et de *B* vers *E*. 229

Tableau 1a : Projets du Software Agents Group au MIT Media Lab (partie 1). 216
Tableau 1b : Projets du Software Agents Group au MIT Media Lab (partie 2). 217

Chapitre V.1

(Pas de figures, ni de graphiques, ni de tableaux.)

CONVENTIONS

Au niveau du style du document, nous suivons certaines conventions. Nous les avons groupées ici.

— Le corps de ce document est composé de *parties* numérotées **I**, **II**, etc.

— Chaque partie contient des *chapitres* numérotés **I.1**, **I.2**, …, **II.1**, **II.2**, etc.

— Chaque chapitre est composé de *sections* numérotées **I.1.1**, **I.1.2**, …, **I.2.1**, **I.2.2**, etc.

— Certaines sections sont composées de *sous-sections* non numérotées.

— La numérotation des *figures*, des *graphiques* et des *tableaux* recommence à 1 dans chaque chapitre.

— Les *références* à des publications classiques sont codées sous la forme [<auteur principal> <année de publication *mod* 100>].

— Les *références* à des pages ou à des sites *Web* sont codées sous la forme [<URL>].

CHAPITRE I.1

LE CONTEXTE

I propose to consider the question, "Can machines think?"
Alan Turing

I.1.1 Introduction

Dans ce premier chapitre, nous présentons le contexte de notre thèse. A partir du contexte large, à savoir l'Informatique, présentée comme discipline scientifique, nous nous approcherons de plus en plus de notre cible, la dynamique organisationnelle des Systèmes Multi-Agents. Nous passerons de l'Informatique aux Agents Autonomes et aux Systèmes Multi-Agents par l'intermédiaire de l'Intelligence Artificielle. Puis, notre aperçu deviendra plus spécialisé et nous discuterons la dynamique organisationnelle même et le problème du maintien de l'intégrité fonctionnelle dans les Systèmes Multi-Agents Ouverts. Nous conclurons ce chapitre en présentant notre vision sur l'Internet, le réseau de plus en plus omniprésent qui constitue à un degré important le contexte de notre thèse.

Notons que ce premier chapitre n'est pas une présentation de l'état de l'art. Le contexte n'est pas le sujet même, mais ce qui se trouve autour. Nous croyons toutefois qu'il est essentiel de le présenter. En effet, nous n'avons pas tous la même vision de ce contexte déterminant la signification de notre modèle et de nos applications. Nous croyons que, en présentant notre vision, une meilleure interprétation de nos travaux sera possible.

Evidemment, ce premier chapitre ne remplace pas la présentation de l'état de l'art. Nous situerons notre modèle et nos applications par rapport aux modèles et aux applications existants après leur présentation. Notre discussion de l'état d'art est donc distribuée sur les parties II, III et IV de cette thèse.

I.1.2 L'Informatique

L'Informatique est un domaine assez particulier. Développée beaucoup plus récemment que la plupart des autres disciplines scientifiques, elle a eu un impact énorme sur tous les aspects de notre vie scientifique et quotidienne. Toutefois, sa nature n'est pas très claire. En particulier, la question se pose de savoir s'il s'agit de science ou d'ingénierie.

Les sciences traditionnelles cherchent à comprendre les phénomènes naturels que nous pouvons observer autour de nous ou provoquer en expérimentant. Les scientifiques essaient d'en construire des modèles correspondant assez fidèlement pour qu'on puisse prédire leur évolution. Par exemple, en observant les mouvements des planètes, nous pouvons construire un modèle et le valider en faisant des prédictions.

Par contre, le but de l'ingénierie traditionnelle est de construire des systèmes artificiels dotés d'une fonctionnalité spécifiée, tout en appliquant des modèles scientifiques validés. Par exemple, en utilisant les modèles de l'aérodynamique, les ingénieurs peuvent concevoir et construire des machines volantes.

Les sciences jouent donc un double rôle. D'un côté, elles permettent la prédiction d'événements. D'un autre côté, elles peuvent nous aider à construire des systèmes utiles. Nous reprendrons la problématique de la *validation* des modèles scientifiques dans le chapitre III.1.

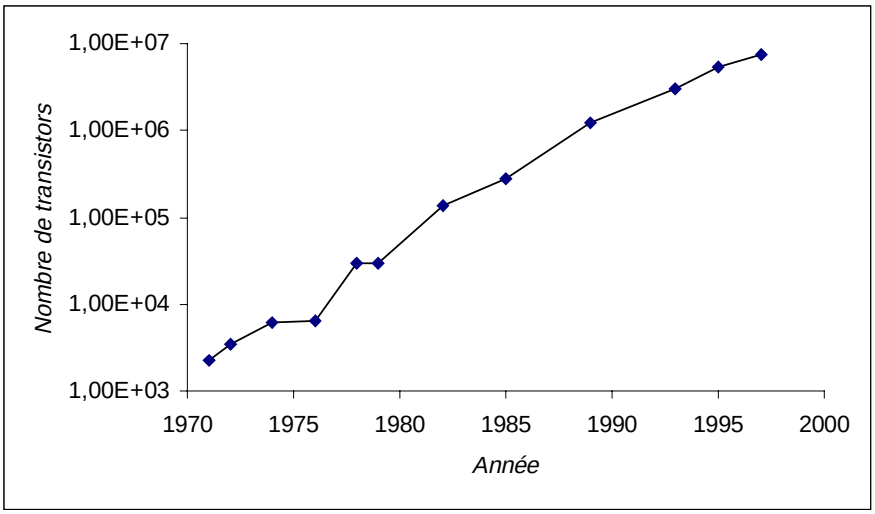
Les premiers modèles scientifiques étaient simples, tout comme les premiers systèmes construits par l'homme. Dans l'évolution de l'homme, ils n'ont pas cessé de croître en complexité. Durant le 20^{ième} siècle, le développement de la micro-électronique et l'informatique a produit une croissance exponentielle de la complexité de systèmes artificiels. Pendant plusieurs décennies, elle suit la loi de Gordon Moore, co-fondateur d'Intel, spécifiant que la complexité (ou capacité, ou performance) des circuits électroniques double chaque 18 mois [Moore]. En 26 ans, le nombre de transistors dans les processeurs d'Intel a augmenté d'un facteur 3200, de 2300 dans le 4004 (1971) jusqu'à 7.5 millions dans le Pentium II (1997). Le tableau 1 et le graphique 1, basés sur des données d'Intel [Intel], montrent cette évolution en détail. Notons que le nombre pour le Pentium Pro n'inclut pas les transistors du *cache*, intégré sur la puce.

Nom de processeur	Date d'introduction	Nombre de transistors
4004	1971	2300
8008	1972	3500
8080	1974	6000
8085	1976	6500
8086	1978	29000
8088	1979	29000
80286	1982	134000
80386	1985	275000
80486	1989	1200000
Pentium	1993	3100000
Pentium Pro	1995	5500000
Pentium II	1997	7500000

Tableau 1 : Evolution du nombre de transistors dans les processeurs d'Intel.

Les microprocesseurs sont des systèmes complexes. Toutefois, ils ne sont que la base pour les systèmes finaux, à savoir des systèmes informatiques composés de hardware *plus* software. En effet, au-dessus du silicium se trouve un système logiciel composé d'un grand nombre de lignes de code. Par exemple, le système de gestion « Windows NT Workstation 4.0 » seul compte déjà plus de 15 millions de lignes de code. De plus, sa taille double - en moyenne - chaque 866 jours [Myhrvold 97]. A ce niveau également, la croissance est exponentielle.

A une échelle mondiale, un seul processeur est un système relativement simple. En effet, l'ensemble des ordinateurs connectés par l'Internet constitue un système artificiel beaucoup plus complexe. Par exemple, si nous estimons - prudemment - la complexité moyenne des hôtes égale à celle d'un 80486 et le nombre d'hôtes à 30 millions, le nombre de composants physiques (transistors) du système mondial est supérieur à 3.6 10^{13}.



Graphique 1 : Evolution du nombre de transistors dans les processeurs d'Intel.

A la différence de machines simples, tel qu'un moteur à explosion ou une horloge mécanique, les systèmes informatiques sont devenus tellement complexes que leur comportement ne peut plus être déduit - d'une manière simple - des comportements des composants individuels. A cause de cette complexité, les systèmes artificiels sont devenus eux-mêmes le sujet de la recherche scientifique. Les chercheurs en Informatique construisent des modèles, permettant - si valides - de prédire leur comportement et leur évolution. Un bon exemple des travaux dans le domaine est la théorie de complexité nous permettant de prédire le temps de calcul d'un programme en fonction de la taille de données. Notons que, si le système est étudié dans sa totalité, comme une boîte noire, l'expérimentation est autant indispensable que dans les autres disciplines scientifiques, bien qu'elle ne soit pas encore très répandue dans le domaine [Tichy 98].

Les modèles développés en Informatique peuvent servir à la prédiction du comportement de systèmes informatiques, mais également à la construction de nouveaux systèmes, fermant la boucle ingénierie → science → ingénierie. Cette boucle, soulignant la relation intime entre les côtés « science » et « ingénierie » de l'Informatique, distingue le domaine des autres disciplines scientifiques. L'Informatique - comme science - étudie un type spécifique de systèmes artificiels et permet la construction de nouveaux systèmes *du même type mais plus complexes*. Ce feed-back positif explique - jusqu'à un certain degré - l'explosion de complexité de ces systèmes.

Les sciences ont toujours étudié les systèmes complexes, sans pourtant incorporer la boucle décrite ci-dessus. Par exemple, la physique statistique étudie des ensembles composés d'un très grand nombre de particules homogènes et en construit des modèles valides. D'autres disciplines ont comme sujet des systèmes composés d'un grand nombre d'éléments *hétérogènes*. Parmi eux, nous trouvons la biologie, la sociologie et l'économie. Les modèles construits au sein de ces disciplines sont d'une nature différente que ceux dans les sciences plus « exactes ». Par exemple, ils ne nous permettent pas la prédiction d'événements économiques tel que les fluctuations des bourses financières. Dans la plupart des cas, ils ne peuvent pas être utilisés pour construire des systèmes pratiques. Les chercheurs engagés dans ces domaines utilisent d'autres critères de validation. Evidemment, à long terme, leurs modèles peuvent s'avérer très utiles selon nos critères aussi.

Nous pouvons constater une coopération de plus en plus forte entre ces domaines et l'Informatique. A un niveau abstrait, les deux parties étudient le même sujet, à savoir des systèmes composés d'un grand nombre de composants hétérogènes. Ce qui les distingue est le fait que les uns étudient des systèmes naturels, et les autres des systèmes artificiels. De l'expertise peut donc être échangée dans les deux directions. De plus, l'Informatique offre un nouvel outil dans l'étude des systèmes complexes naturels : la *simulation*.

I.1.3 L'Intelligence Artificielle

Si l'Informatique est une science exceptionnelle et difficile à situer, l'Intelligence Artificielle (IA) l'est encore plus. Ses débuts, comme discipline scientifique, se situent au milieu des années 50, quelques années après l'invention de l'ordinateur dans sa forme actuelle par von Neumann et ses collègues [von Neumann 45]. Pendant le Darthmouth Conference, rassemblant John McCarthy, Marvin Minsky, Nathaniel Rochester et Claude Shannon, le terme *Artificial Intelligence* était utilisé pour la première fois [McCarthy 55]. L'hypothèse à la base de la conférence était « …the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. » [ibid.].

C'est un fait à la fois étonnant et fascinant qu'un des premiers phénomènes que l'homme a essayé de simuler était l'intelligence humaine. Sans doute, le fait qu'il semble s'agir d'un phénomène purement mental, et non physique, a joué un rôle dans ce choix. Toutefois, depuis le début, les ambitions de l'Intelligence Artificielle étaient formidables, vu la simplicité des ordinateurs et des modèles existants et la complexité de l'être humain. La proposition originale de McCarthy et al. incluait la simulation de processus tels que l'intuition et la création musicale. On voulait reproduire tous les aspects de l'intelligence, jusqu'à ce que le système artificiel soit indiscernable d'un être humain. En effet, une des premières méthodes d'évaluation proposées pour des entités artificiellement intelligentes était le Test de Turing dans lequel l'évaluateur essaie, en utilisant une interface textuelle, de distinguer une personne d'un système informatique [Turing 50].

Pendant ces 50 dernières années, nous avons beaucoup appris à propos du problème de l'Intelligence Artificielle. Surtout, nous avons compris qu'il s'agit d'un problème très difficile. Ceci a motivé un grand nombre de chercheurs en IA à modifier les ambitions et les perspectives de la discipline. Le but initial s'est peu à peu réduit à la construction de systèmes assistant l'homme dans des tâches nécessitant de l'intelligence ou à la résolution de problèmes en général. Nous ne partageons pas cette vision limitée, et restons fidèles aux ambitions originales. Ceci n'implique pas que nous voulons construire une machine capable de passer le Test de Turing demain. Par contre, ceci implique que nous voulons comprendre des *principes de base* qui peuvent nous mener - à long terme - à une telle réalisation.

Les travaux de cette thèse sont effectués dans cet esprit, mais - comme nous le verrons - ils se situent à un niveau très abstrait. Contrairement à ce qui est habituel dans une grande partie du domaine, ils ne concernent pas directement des concepts cognitifs tels que des « croyances » ou des « intentions ». Dans un premier temps, notre but est la compréhension maximale d'un modèle minimal de ce que nous croyons indispensable pour réaliser des systèmes artificiellement intelligents.

L'Intelligence Artificielle est une discipline particulière, parce qu'elle est directement liée à un phénomène naturel (l'intelligence), toutefois sans vouloir le modéliser. En effet, son but principal est la construction de systèmes reproduisant le phénomène, plutôt que sa modélisation analytique. Ceci distingue le domaine nettement

de la Psychologie et des Sciences Cognitives, des disciplines souhaitant modéliser l'esprit humain, éventuellement avec l'aide d'ordinateurs. Evidemment, une fois le phénomène reproduit (dans les marges voulues), le système informatique peut servir comme modèle scientifique. Toutefois, soulignons que la philosophie de l'Intelligence Artificielle, comme elle a été définie par les fondateurs, est orientée vers la construction de systèmes artificiels, ce qui place la discipline au sein de l'Informatique. Du point de vue scientifique, l'Intelligence Artificielle implique la modélisation de systèmes informatiques, plutôt que la modélisation de systèmes naturels. Comme l'illustre la figure 1, nous voulons créer et raffiner des **modèles informatiques**.

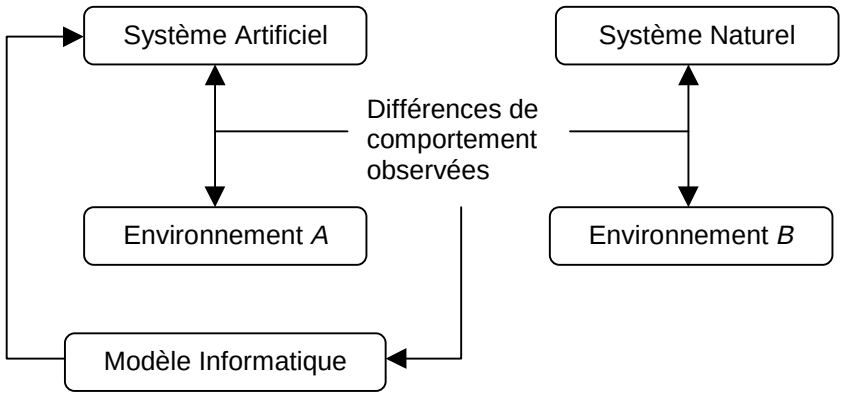


Figure 1 : Le rôle du modèle informatique dans l'Intelligence Artificielle.

Ce modèle informatique est utilisé pour construire le système artificiel. En laissant interagir le système avec un environnement donné, nous pouvons observer son comportement et le comparer au comportement du système naturel intégré dans son propre environnement. Puis, nous pouvons - en interprétant les différences observées - raffiner notre modèle et répéter le cycle. Evidemment, cette approche n'est valide que lorsque les deux environnements (étiquetés *A* et *B* dans la figure 1) sont similaires, et idéalement identiques. Toutefois, pour des raisons pratiques, *A* est souvent d'une nature différente de *B*. Comme nous le verrons dans la section suivante, la question du degré de similarité nécessaire est à la base d'une controverse importante dans le domaine.

Concluons cette section en soulignant que l'intelligence semble être un phénomène très complexe. Elle paraît être liée à tous les aspects de l'esprit humain et l'approche réductrice classique semble être insuffisante pour en construire un modèle valide. De plus, le phénomène naturel est dépendant d'un substrat physique d'une complexité impressionnante. En effet, le cortex humain est composé de milliards de neurones, tous d'une complexité considérable [Koch 97]. A cause de ces raisons, nous croyons que l'approche constructrice de l'Intelligence Artificielle, couplée au progrès incessant de l'Informatique, est une des seules pistes vers une meilleure compréhension du phénomène.

I.1.4 Les Agents Autonomes

Un de premiers systèmes construits dans l'Intelligence Artificielle était le *Logic Theorist*, développé par Alan Newell et Herbert Simon en 1955 [Crevier 93]. Le nom du programme suggère une certaine vision anthropomorphe : le système est présenté comme un agent ayant une certaine autonomie. Cette vision des choses était typique pour les premières 10, 20 années de la discipline. Toutefois, l'autonomie des ses systèmes était très

relative, surtout si on tient compte du fait qu'ils étaient tous très spécialisés. Ils étaient plutôt des « cerveaux artificiels » implémentant des fonctions supérieures typiquement associées à l'homme, tels que le raisonnement, l'apprentissage et la traduction automatique de langues naturelles.

La spécialisation de l'IA

Les pionniers de l'IA étaient enthousiastes et optimistes sans bornes. Par exemple, Marvin Minsky, fondateur du laboratoire pour l'Intelligence Artificielle au MIT, prédisait en 1967 que « within a generation … the problem of creating 'artificial intelligence' will be substantially solved » [ibid.]. Cet optimisme naïf était le carburant pour le discours des critiques de l'IA tel que Hubert Dreyfus. Son rapport défavorable « Alchemy and Artificial Intelligence » [Dreyfus 65], plus tard développé dans le livre « What Computers Can't Do: A Critique of Artificial Reason » [Dreyfus 93], cite à volonté les affirmations de ces chercheurs.

En effet, l'optimisme n'était pas fondé. Par exemple, les programmes informatiques n'arrivaient pas a acquérir le *sens commun* : la signification de simples histoires pour enfants leur échappait. Dans le monde physique, la navigation dans des espaces naturels s'avérait un problème trop compliqué pour les robots incorporant les algorithmes IA.

Ces problèmes motivaient les chercheurs en IA pour simplifier les environnements dans lesquels les entités artificielles opéraient. Au niveau virtuel, le concept du *micro-monde* était né : des environnements tellement spécialisés que le problème du sens commun ne se posait plus. Au niveau physique, les environnements des robots étaient spécialement adaptés pour leur rendre la vie moins compliquée. Au lieu d'approcher les environnements *A* et *B* du modèle présenté dans la section précédente, les chercheurs les éloignaient. En conséquence, la quête pour l'intelligence artificielle s'était transformé en l'étude de la résolution de problèmes isolés.

La renaissance de l'IA

Restreinte à des domaines limités, l'IA a produit des résultats impressionnants. Un des exploits les plus remarquables est sans doute la victoire de Deep Blue sur Garry Kasparov. Le 11 mai 1997, Deep Blue, une machine massivement parallèle, développée par IBM, battait Karparov, champion de monde, aux échecs [Deep Blue]. Ceci était un grand moment pour l'IA, même si les développeurs soulignaient que leur système n'utilisait pas de l'intelligence artificielle [Deep Blue FAQ].

En effet, Deep Blue ne faisait que résoudre un problème dans un univers minimal, et la vraie intelligence implique l'autonomie de systèmes dans un environnement ouvert, dynamique et pas toujours prévisible. Cette absence d'autonomie était la critique principale du domaine, exprimée - entre autres - par Rodney Brooks à partir de 1985. Ce chercheur notoire, travaillant à MIT, avertissait explicitement la communauté qu'elle s'était éloignée loin de son but original. Comme l'indique le titre de son papier « Elephants Don't Play Chess » [Brooks 90a], il argumentait que la résolution de problèmes abstraits ne suffit pas pour garantir l'autonomie de vrais systèmes.

Brooks et des chercheurs partageant sa vision ont eu une influence très importante sur le domaine de l'IA et sont à la base de toute une nouvelle école. Toutefois, notons qu'un grand nombre de chercheurs, dans le passé et dans

le présent, sont toujours très sceptiques à propos de ses idées. Les idées essentielles de la nouvelle école sont les suivantes :

— Le thème central de l'IA doit être la création d'agents autonomes, plutôt que la résolution de problèmes hors contexte.

— Les systèmes artificiellement intelligents doivent opérer dans les même environnements que les systèmes naturellement intelligents.

— Le raisonnement et la représentation symboliques sont à éviter.

— Par conséquent, il nous faut des architectures réactives, plutôt que cognitives.

— Une architecture réactive possible est la *subsumption architecture*.

Etudions et évaluons - en quelques lignes - ces idées, que Brooks a exposées dans des articles variés [Brooks 85-98].

Autonomie

L'école de Brooks argumente que l'intelligence naturelle est directement liée à l'autonomie. En effet, tout système naturel ayant des propriétés associées à l'intelligence est autonome dans son environnement. Il est difficile de donner une définition exacte de l'autonomie, mais la notion implique généralement que le système peut opérer indépendamment d'une entité superviseur et qu'il contrôle - jusqu'à un certain degré - son propre destin.

Essayons de trouver une définition pertinente dans un domaine où l'autonomie n'est pas seulement désirable, mais d'une importance cruciale. L'exploration de l'espace, par exemple, est basée à un degré important sur des systèmes autonomes. L'édition « Autonomy in Space » (septembre / octobre 1998) du journal *IEEE Intelligent Systems* [Doyle 98] est dédié à cette problématique. Les articles dans cette édition nous présentent le point de vue des chercheurs des instituts tels que JPL et NASA. Le consensus général est exprimé par Crystal Chweh, qui nous propose : « A loose definition gives autonomy as a self-regulating, self-controlling system. » [Chweh 98].

Robert Rasmussen, du JPL, y ajoute l'observation suivante : « In the final analysis, autonomy isn't so much a line you cross as it is a goal you never quite achieve. » [Rasmussen 98]. Cette phrase exprime le fait que l'intelligence et l'autonomie sont des buts ultimes et que leur définition évolue avec notre progrès. L'Intelligence Artificielle est une science orientée vers le futur, ce qui explique et justifie - jusqu'à un certain degré - les projections optimistes dans le domaine : elles nous motivent pour continuer. Autour du domaine, nous trouverons toujours des futuristes tels que Hans Moravec, offrant des discours spéculatifs, mais propices à l'inspiration [Moravec 90].

Nous sommes convaincus qu'un système intelligent doit être autonome dans le sens proposé ci-dessus. Notons que, pour certains chercheurs, un *système autonome* constitue une base nécessaire, mais pas suffisante d'un *agent autonome*. Comme nous le verrons vers la fin de cette section, le terme « agent » implique souvent d'autres propriétés.

Le monde physique

Il est essentiel pour Brooks, roboticien, que les environnements des systèmes IA soient physiques. Pour lui, les environnements *A* et *B* dans notre modèle doivent être identiques, impliquant que les systèmes artificiellement intelligents doivent être des robots physiques.

Un certain nombre de chercheurs ont argumenté qu'il suffit que l'environnement d'un système IA soit de la même nature qu'un environnement physique, à savoir *dynamique*, *bruyant* et *imprévisible*. Un exemple excellent d'un tel environnement est l'Internet, un espace riche pour des entités virtuelles.

Nous partageons ce dernier point de vue, défendu notamment par Oren Etzioni [Etzioni 93] [Etzioni 94] [Etzioni 95]. L'identité des environnements *A* et *B* est un but ultime. Surtout pour des raisons pratiques, il nous semble qu'il est permis de viser - dans un premier temps - des buts plus modestes.

Intelligence sans raisonnement, sans représentation

Le *Physical Symbol System Hypothesis* (PSSH), formulé par Newell et Simon, est à la base de l'Intelligence Artificielle classique. Selon cette hypothèse, un système utilisant un ensemble de symboles, physiquement associés, et un ensemble d'opérateurs définis sur ces symboles a les moyens nécessaires et suffisants pour l'action intelligente générale [Newell 80]. Nous pouvons formuler l'essentiel de cette hypothèse - d'une manière formelle - comme suit :

$$S^{-1}(S(T)[(S(X)]) = T[X]$$

Dans cette formule, X représente une entité réelle, $S(X)$ sa représentation symbolique (donc, un symbole), $T[\ ]$ une transformation réelle de X, et $S(T)$ la représentation symbolique de cette transformation. $S^{-1}(\ )$ représente l'inverse de $S(\ )$. La formule implique que le monde symbolique peut être un miroir du monde réel. Notons que nous avons utilisé l'adjectif « réel », plutôt que « physique » pour qualifier X et T. En effet, il ne s'agit pas seulement d'objets physiques, mais également d'entités virtuelles. Par exemple, X peut être un état émotionnel, ou un événement ou un contrat.

Dans l'IA classique, la plupart des travaux sont effectués dans l'espace symbolique. Toutefois, dans le contexte de la PSSH, ces travaux ne sont utiles que si nous avons accès aux fonctions $S(\ )$ et $S^{-1}(\ )$. Pour un grand nombre de chercheurs dans l'IA, ceci était un détail. Par exemple, dans un contexte robotique, il était considéré comme un problème technique impliquant des capteurs et des moteurs. Toutefois, la communauté n'arrivait pas à construire des systèmes complets, comme des robots, démontrant de l'intelligence artificielle, et tout indiquait que la réalisation des deux fonctions constituait un problème difficile et fondamental. Stevan Harnad le baptisait le *Symbol Grounding Problem* (SGP) [Harnad 90], John Searle l'utilisait comme arme contre l'IA [Searle 80] et Rodney Brooks en faisait un thème central dans son discours.

Illustrons le SGP à l'aide d'un exemple. Dans l'homme, la notion « il pleut » est liée à une multitude d'expériences sensorielles, de réflexions et de souvenirs. Harnad dit que la notion est *enracinée* dans notre corps et dans notre environnement physique. Par contre, est-ce que c'est le cas du symbole « il pleut » dans un programme de l'IA classique ? Est-ce qu'il représente vraiment l'événement, ou est-ce que c'est juste une chaîne de bits, n'ayant une signification que dans les yeux des programmeurs ?

Certains chercheurs en IA classique ont essayé de résoudre le problème en construisant - d'une manière systématique et artificielle - des réseaux complexes de symboles, espérant que - une fois la masse critique atteinte - l'intelligence émergera. C'est notamment le cas du projet « CYC » de Doug Lenat qui espère créer des systèmes ayant du *sens commun* en compilant une base de données encyclopédique [Cyc]. Ces systèmes n'incorporent pas la notion d'enracinement, qui nécessite une interaction intense et continue entre les systèmes et leur environnement. Jusqu'à maintenant, ils n'exhibent pas de comportement réellement intelligent. Selon l'école de Brooks, ils ne l'exhiberont jamais.

La solution proposée par Brooks au SGP était radicale : éviter toute manipulation symbolique, et ne construire que des systèmes *réactifs*.

Nous aussi, nous sommes convaincus que la réalisation des fonctions $S(\ )$ et $S^{-1}(\ )$ pose un véritable problème. Toutefois, il ne nous semble pas indiqué d'interdire des manipulations symboliques, car *elles sont inévitables dans un ordinateur numérique*. En effet, même au niveau purement numérique, un ordinateur numérique ne manipule pas des nombres, mais des représentations symboliques de nombres. Donc, au lieu d'éviter les fonctions $S(\ )$ et $S^{-1}(\ )$, essayons plutôt de mieux les comprendre et de tenter leur réalisation.

Dans un premier temps, nous avons essayé de trouver un monde (environnement *A*), dont les éléments se traduisent relativement facilement en symboles et vice versa. Dans cette thèse, nous proposons un monde basé sur les notions de **couple** et de **similarité entre entités**. Nous croyons que ces notions se traduisent facilement en symboles et vice versa.

Le monde que nous proposons est très simple comparé avec notre monde physique (environnement *B*). Si nous voulons réaliser des entités artificiellement intelligentes, *A* doit être suffisamment proche de *B*. Nous croyons que, le degré de complexité exclu, *A* et *B* ont beaucoup en commun. Dans le cadre de cette thèse, nous ne pouvons que suggérer cette similarité. En effet, une étude profonde de cette hypothèse impliquerait - entre autre - la coopération avec des physiciens.

Concluons cette sous-section en soulignant que nous considérons le SGP comme essentiel et que la nature même de notre modèle en est inspirée. Toutefois, dans cette thèse, nous nous concentrerons sur le modèle même, plutôt que sur son rôle dans l'Intelligence Artificielle en général.

Les architectures réactives

Les architectures *réactives* évitent la manipulation de symboles des architectures *cognitives*. Elles s'appuient sur une interaction intense entre le système et son environnement. Il n'y pas de représentation explicite de l'environnement : *soit* elle est implicite, *soit* elle est complètement évitée. Il n'y pas de raisonnement non plus : le comportement intelligent est émergent des interactions. Comme exemples de ce type d'architecture, nous trouvons les réseaux de neurones, certains algorithmes génétiques et l'architecture « subsumption » de Brooks. Un terme souvent plus correct pour désigner ces architectures est « sous-cognitives », plutôt que « réactives ».

Partageant l'opinion d'un nombre croissant de chercheurs en IA, nous croyons que ni les architectures réactives, ni les architectures cognitives offrent la solution unique. En fonction du contexte, certaines architectures sont plus adaptées que d'autres. Dans cette thèse, nous faisons abstraction de la distinction « réactif - cognitif ». Notre modèle est compatible avec les deux types d'architecture.

L'architecture « subsumption »

Rodney Brooks a proposé sa propre architecture réactive, adaptée pour des robots autonomes, et baptisée l'architecture « subsumption ». Le principe de base est illustré par la figure 2. Dans les systèmes classiques, l'architecture est typiquement composée d'un nombre de modules traitant - d'une manière séquentielle - les signaux venant des capteurs. A la fin de la chaîne, des signaux pour des moteurs sont générés. Par exemple, une telle architecture peut être composée d'un module « perception », un module « traitement », et un module « action ».

Brooks propose une décomposition alternative, basée sur l'identification d'un certain nombre de comportements (*behaviors*) du système. Au niveau architectural, à chaque comportement correspond un module informatique. Dans les robots autonomes, des exemples de comportements typiques sont « avancer », « éviter obstacles », et « recharger ». Chaque module inclut la chaîne complète de traitement, allant de la perception à l'action. Dans la figure 2, la décomposition verticale correspond aux architectures classiques, la décomposition horizontale à l'architecture de Brooks.

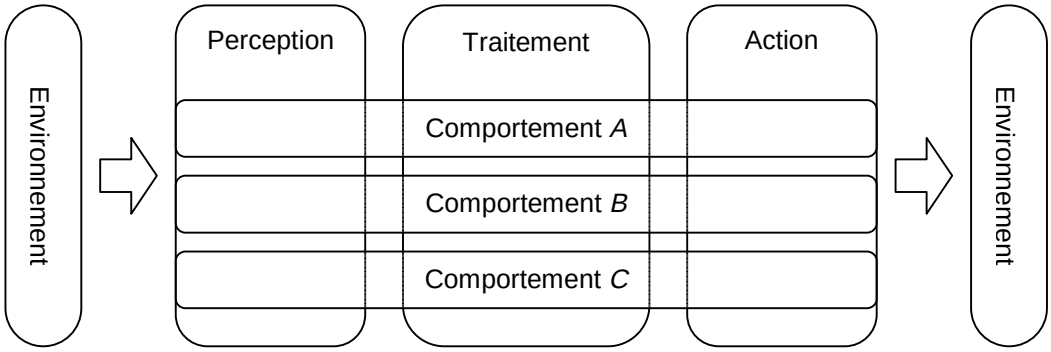


Figure 2 : La décomposition horizontale et verticale.

En général, les comportements sont organisés selon des niveaux différents, de très primitifs (par exemple, « avancer ») à plus sophistiqués (par exemple, « prendre une canette de Coca »). Le comportement global du système, qui peut être relativement sophistiqué, est le résultat des interactions entre les modules différents. Comme l'indique le nom de l'architecture, les interactions de base consistent en la « subsumption » de certains modules par d'autres, typiquement à un niveau supérieur.

Comme notre modèle fait abstraction de l'architecture des agents, cet aspect des travaux de Brooks a peu d'importance pour notre thèse.

La cybernétique

Beaucoup a été écrit sur la « renaissance » de l'Intelligence Artificielle et on peut argumenter qu'elle a été sur-médiatisée. Toutefois, nous croyons qu'elle a joué un rôle essentiel en redirigeant la discipline vers son but original. Notons pourtant que les idées de l'école de Brooks ne sont pas nouvelles.

Déjà au début des années 50, W.G. Walter construisait des robots très similaires aux créatures de Brooks [Walter 50] [Walter 51]. En général, au sein de la discipline de la *cybernétique*, des chercheurs comme Norbert Wiener

[Wiener 48] et Heinz von Foerster [von Foerster 74] développaient des idées liées, réfutées pendant longtemps comme « unproductive » [Varela], mais maintenant resurgissantes dans le domaine de l'Intelligence Artificielle.

L'invasion des agents

Le concept d'agent n'est pas nouveau dans l'IA. Comme nous l'avons indiqué, déjà au *Logic Theorist* de 1955 était associé la notion d'agent et une certaine forme d'autonomie. Toutefois, il s'agissait de programmes résolvant des problèmes, plutôt que des entités autonomes dans le sens de l'école de Brooks.

Les années 90 ont été caractérisées par une véritable explosion de travaux sur les agents autonomes. Les travaux de Brooks ont sans doute - jusqu'à un certain degré - stimulé ce développement, mais - en général - les modèle proposés ne sont pas conçus dans le même esprit. En particulier, la plupart des agents construits sont virtuels, plutôt que physiques : il s'agit de *software agents*. Il existe une littérature extensive sur le sujet, et plusieurs œuvres présentent un aperçu du domaine. Trois collections générales et récentes d'articles sont [Bradshaw 96], [Huhns 98] et [Jennings 98a].

En général, les chercheurs dans le domaine des agents s'appuient sur le Physical Symbol System Hypothesis, bien que ses défauts aient été soulignés par un grand nombre de scientifiques. Une raison plausible pour ce phénomène est donnée par Wooldridge et Jennings : « The deliberative, symbolic paradigm is, at the time of writing, the dominant approach in (D)AI. This state of affairs is likely to continue, at least for the near future. There seem to be several reasons for this. Perhaps most importantly, many symbolic AI techniques (such as rule-based systems) carry with them an associated technology and methodology that is becoming familiar to mainstream computer scientists and software engineers. Despite the well-documented problems with symbolic AI systems, this makes symbolic AI agents […] an attractive proposition when compared to reactive systems, which have as yet no associated methodology. » [Wooldridge 95].

Indépendamment de ces raisons, ce phénomène implique qu'un grand nombre de chercheurs utilisent comme méthode d'implémentation des méthodes déclaratives, adaptées à la description externe de systèmes (par un observateur). Par exemple, l'*intentional stance* de Daniel Dennett réfère à la *description* d'un agent en y *attribuant* des états intentionnels, et de la rationalité [Dennett 87]. Toutefois, il est adopté comme méthode d'*implémentation* par un nombre considérable de chercheurs. Par exemple, l'architecture « Beliefs, Desires, Intentions (BDI) » est directement liée à cette méthode descriptive [Rao 91].

Pour des raisons peu documentées, ce groupe de chercheurs semble penser que « l'animation » de ces descriptions déclaratives, à l'aide de l'informatique, produira les systèmes que nous voulons créer : des agents réellement autonomes. Des problèmes tels que le *Symbol Grounding Problem* sont typiquement niés.

Notons cependant que certains chercheurs, travaillant sur ces modèles symboliques, sont très conscients de leurs limitations et n'hésitent pas à les citer. Wooldridge et Jennings, par exemple, affirment que - pour eux - ces théories ne servent qu'à *spécifier* des agents. Les deux chercheurs n'offrent pas de mécanismes généraux pour traduire ces spécifications en des implémentations, mais affirment qu'il s'agit d'un problème difficile [Wooldridge 95].

La définition de la notion d'agent

Comme l'illustre la controverse décrite ci-dessus, des opinions très différentes coexistent dans le domaine des Agents Autonomes. En conséquence, il est peu surprenant de trouver un grand nombre de définitions différentes de la notion d'agent.

Stan Franklin et Art Graesser parcourent dans [Franklin 96] une douzaine de définitions et en distillent la définition suivante : « An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future. » Il s'agit d'une définition relativement large, couvrant un grand nombre de systèmes.

Jacques Ferber nous donne, dans [Ferber 95], une définition beaucoup plus détaillée :

« On appelle agent une entité physique ou virtuelle

a. qui est capable d'agir dans un environnement,
b. qui peut communiquer directement avec d'autres agents,
c. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
d. qui possède des ressources propres,
e. qui est capable de percevoir (mais de manière limitée) son environnement,
f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
g. qui possède des compétences et offre des services,
h. qui peut éventuellement se reproduire,
i. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit. »

Cette deuxième définition est assez proche de la première. Toutefois, elle apporte deux nouveaux éléments importants ((h) exclu, étant optionnel). Primo, Ferber identifie la capacité de communication (b). Ceci est peu surprenant, car ses travaux se situent dans le domaine des Systèmes *Multi*-Agents (voir la section suivante). Si on veut inclure les agents réactifs dans cette définition, la notion de communication doit être interprétée dans le sens large d'interaction (et pas dans le sens limité de communication symbolique). Secundo, la définition de Ferber implique qu'un agent offre des services (g), liant la notion d'agent à la notion d'utilisateur.

Ce dernier élément lie la définition de Ferber aux travaux de Pattie Maes, une des pionnières dans le domaine des *software agents*. La description du Software Agents Group, qu'elle a créé au début des années '90 au MIT Media Laboratory, est la suivante : « The Software Agents Group of the MIT Media Laboratory investigates computer systems to which one can delegate tasks. Software agents differ from conventional software in that they are long-lived, semi-autonomous, proactive, and adaptive. The group develops techniques and builds prototype agent systems that can be tested. » [software agents]. Cette description illustre la vision de Maes, dans laquelle les *utilisateurs* des agents jouent un rôle central. Dans son article séminal « Agents that Reduce Work and Information Overload », elle présente les agents comme des *assistants personnels* (*personal assistants*), *collaborant* avec les utilisateurs et partageant leur environnement de travail [Maes 94].

Dans cette thèse, nous prenons un point de vue large et abstrait de la notion d'agent. Dans le chapitre II.2, nous définirons un agent comme un *processus*, i.e. quelque chose qui est actif, doté d'une certaine *autonomie*, i.e. qui n'est pas complètement contrôlé par un autre processus.

I.1.5 Les Systèmes Multi-Agents

Pendant les premières vingt années de l'Intelligence Artificielle classique, les systèmes développés étaient, comme le *Logic Theorist*, solitaires. Les chercheurs construisaient des « cerveaux artificiels » isolés et les confrontaient aux problèmes tels que les échecs ou la démonstration de théorèmes mathématiques. Dans l'esprit général, la dimension culturelle de l'intelligence, impliquant nécessairement une dimension sociale, était niée.

Toutefois, certains problèmes étudiés, tels que le traitement automatique des langues, avaient une dimension culturelle évidente. De plus, le Turing Test même, à l'époque non encore considéré comme très difficile, est très dépendant de facteurs culturels. Comment cette dimension a-t-elle pu être négligée ?

L'Intelligence Artificielle classique était un domaine de spécialistes en sciences « dures » comme les mathématiques ou la physique. De plus, ces chercheurs s'étaient concentrés sur le côté technique de la problématique. Non par hasard, le premier laboratoire spécialisé en intelligence artificielle était celui fondé par Marvin Minsky au Massachusetts Institute of Technology. La puissance fascinante, et peut-être aveuglante, de l'ordinateur numérique, qu'ils venaient de découvrir, et la nature de leur spécialisation jouaient dans doute un rôle dans leur vision « mono-agent ». A un niveau technique, la notion de réseau informatique était virtuellement inexistante : il y avait à peine des ordinateurs. De plus, le concept de computation distribuée était plus ou moins abandonné, bien que les réseaux de neurones aient toujours été un sujet de recherche pour Minsky même (voir, par exemple, [Minsky 54]). Notons que les contributions de Minsky à ce domaine, bien que de grande importance, soulignaient certaines limitations de l'approche [Minsky 69] et ont sans doute retardé son développement. Donc, au niveau technique également, les systèmes isolés étaient au centre des recherches.

Par contre, vers la fin des années 70, des systèmes distribués, au niveau conceptuel et au niveau technique, sont apparus dans l'Intelligence Artificielle. Notamment les concepts de « tableau noir » et d'« acteur » étaient introduits par des chercheurs comme Frederick Hayes-Roth, Victor Lesser et Carl Hewitt [Erman 80] [Lesser 83] [Hewitt 77]. Les systèmes étaient conçus dans l'esprit de l'IA classique. Les agents individuels étaient conçus comme des entités cognitives et leur but était la résolution distribuée de problèmes. En conséquence, une partie importante des recherches dans ce domaine, appelé « Distributed Artificial Intelligence » (DAI), peut être classifiée comme « Distributed Problem Solving » (DPS).

A partir des années 80, le domaine n'a pas cessé de croître. Un bon aperçu des recherches jusqu'à 1988 est donné par [Bond 88] ; une présentation plus récente est [O'Hare 96]. Pendant sa croissance, la vision du domaine est devenue plus large. En particulier, un nombre important de chercheurs a intégré le point de vue et les méthodes des courants plus récents, tel que l'école de Brooks. Dans ces recherches plus récentes, les chercheurs utilisent non seulement des agents cognitifs, mais également réactifs. Si possible, ils font abstraction de l'architecture des agents et se concentrent sur les **interactions** entre agents. Au niveau des applications également, ils prennent un point de vue différent : il n'est plus nécessaire que les systèmes résolvent des problèmes cognitifs, il suffit que leur comportement puisse être utile dans un contexte plus large, comme la

modélisation ou la simulation. La différence entre la DAI classique et le domaine plus large et plus récent est reflétée dans son nom, à savoir le domaine des **Systèmes Multi-Agents** (SMA).

La DAI classique était un domaine dominé par des chercheurs américains. Le domaine des SMA, par contre, est une affaire internationale, rassemblant - entre autres - des chercheurs américains, européens et japonais. Il ne s'agit pas d'un phénomène de « jumping on the bandwagon ». Au contraire, les contributions européennes ont été déterminantes dans la définition du domaine actuel. En France, Yves Demazeau et Jacques Ferber peuvent être considérés comme les co-fondateurs du domaine. Yves Demazeau, dès le début très actif dans les *workshop* MAAMAW (*Modeling Autonomous Agents in a Multi-Agent World*) [Demazeau 90] [Demazeau 91], a récemment présidé la troisième ICMAS (*International Conference on Multi-Agent Systems*) à Paris [Demazeau 98]. Jacques Ferber était le premier, à une échelle mondiale, de publier une monographie sur ce nouveau domaine que sont les Systèmes Multi-Agents [Ferber 95].

Notons que le livre de Ferber donne un bon aperçu du domaine actuel. Une description plus compacte est donnée dans l'éditorial du premier numéro d'*Autonomous Agents and Multi-Agent Systems*, un journal récent d'une orientation explicitement internationale [Jennings 98b].

Un certain nombre de chercheurs dans le domaine, la plupart européens, étudient des SMA composés uniquement d'agents réactifs. Dans ce cas, les interactions entre les agents sont sous-cognitives, puisque les agents n'ont pas la capacité de manipuler des symboles. La fonctionnalité d'un tel système n'est pas codée explicitement (d'une manière symbolique) dans le système même, mais elle en *émerge*. On parle de la fonctionnalité émergente. Ce phénomène est étudié - entre autres - par Luc Steels, Jacques Ferber et Alexis Drogoul [Steels 91] [Ferber 92] [Drogoul 93]. La granularité de ces systèmes est typiquement importante : ils comptent un grand nombre de simples agents. A ce niveau, le domaine des SMA est très proche du domaine de la Vie Artificielle, cette fascinante discipline, représentée - entre autres - par Christopher Langton [Langton 88].

Notons toutefois qu'il existe une différence essentielle entre les deux disciplines : l'une étudie l'intelligence (artificielle), l'autre étudie la vie (artificielle). A quel degré ces deux problématiques sont liées est une question scientifique importante, mais difficile. Luc Steels nous offre des réflexions intéressantes dans [Steels 94] et [Steels 95]. Nous présenterons notre vision de cette problématique, périphérique à notre thèse, vers la fin du chapitre II.5.

Notre thèse se situe dans le domaine des Systèmes Multi-Agents. En particulier, nous proposons un modèle qui peut être utile dans l'étude de la dynamique organisationnelle dans ces systèmes. Comme nous l'avons indiqué dans la section précédente, nous avons une vision large et abstraite de la notion d'agent. Notre vision du concept des SMA est de la même nature. Dans le chapitre II.2, nous définirons, tout simplement, un SMA comme un système composé de plusieurs agents. En conséquence, notre modèle peut aider à l'étude de virtuellement tout système dans les domaines des SMA et de la Vie Artificielle. Par contre, à cause de sa généricité, il n'existe pas de « niche » scientifique pour notre modèle, et il est difficile de le situer dans un des sous-domaines étudiés par les chercheurs en SMA.

I.1.6 La dynamique organisationnelle des Systèmes Multi-Agents

Lorsqu'on assemble plusieurs agents la question se pose de savoir comment les organiser. Par conséquent, la notion d'organisation joue un rôle essentiel dans les SMA. L'organisation d'un système peut être dynamique. Dans ce cas, l'étude de sa dynamique organisationnelle s'impose.

La notion d'organisation est intensivement étudiée dans le domaine. Par exemple, Yves Demazeau identifie, dans sa méthodologie « voyelles », quatre dimensions essentielles des SMA : l'agent (A), l'environnement (E), l'interaction (I) et l'organisation (O) [Demazeau 97]. Jacques Ferber indique que « L'organisation est, avec l'interaction, l'un des concepts de base des systèmes multi-agents. » [Ferber 95].

Toutefois, il s'agit d'une notion complexe et difficile à définir. Le mot « organisation » a plus d'un sens. Dans les termes « OTAN » ou « organisation criminelle », la notion indique un Système Multi-Agents concret. Nous n'utiliserons pas le terme « organisation » dans ce sens, nous parlerons de « système ». Puis, la notion indique l'agencement de relations entre agents, un concept abstrait. En effet, on peut parler de « l'organisation de l'OTAN ». Nous utiliserons le terme dans ce sens.

Notons qu'il existe une relation intime entre un système et son organisation. La disposition des relations entre agents détermine l'évolution future du système. Puis, le système même détermine l'évolution de son organisation. Par exemple, les relations entre agents, suivant le modèle d'une hiérarchie, déterminent l'évolution d'une unité militaire. Toutefois, c'est l'unité même qui génère (soutient) la hiérarchie : les deux composants ne peuvent pas exister indépendamment. Cette boucle entre système et organisation peut également exister à un niveau supérieur. Par exemple, une bonne unité militaire respectera toujours le modèle d'une hiérarchie, indépendamment de l'évolution des agents et des relations entre agents. Le *type* d'organisation détermine comment l'organisation concrète peut évoluer. Dans l'autre sens, l'organisation concrète détermine si le système reste dans une *classe* donnée d'organisations possibles. Certains auteurs appellent une telle classe également « organisation ». Par exemple, dans un contexte biologique, Humberto Maturana et Francisco Varela appellent un système « vivant » lorsqu'il arrive à maintenir son organisation dans ce dernier sens [Maturana 80]. Nous reprendrons cette problématique dans le chapitre II.5. La figure 3 montre les boucles entre les niveaux d'organisation différents.

Les sens différents du terme « organisation » peuvent provoquer des malentendus, surtout parce que la terminologie développée pour les distinguer n'est pas normalisée. Dans le tableau 2, nous présentons la nôtre à côté de celle utilisée par Ferber et Maturana. Nous avons introduit le terme « signature organisationnelle » pour éviter la confusion entre les termes « structure » et « organisation ». Dans notre terminologie, ces deux derniers termes sont synonymes.

Dans cette thèse, nous nous concentrons sur le niveau intermédiaire, indiqué par un style italique dans le tableau 2. De plus, nous nous limitons à un type de relations en particulier, à savoir des **relations de groupement**. Ceci est une différence principale entre notre approche et l'approche classique étudiant des relations à un niveau plus haut, telles que des relations fonctionnelles ou des relations de pouvoir.

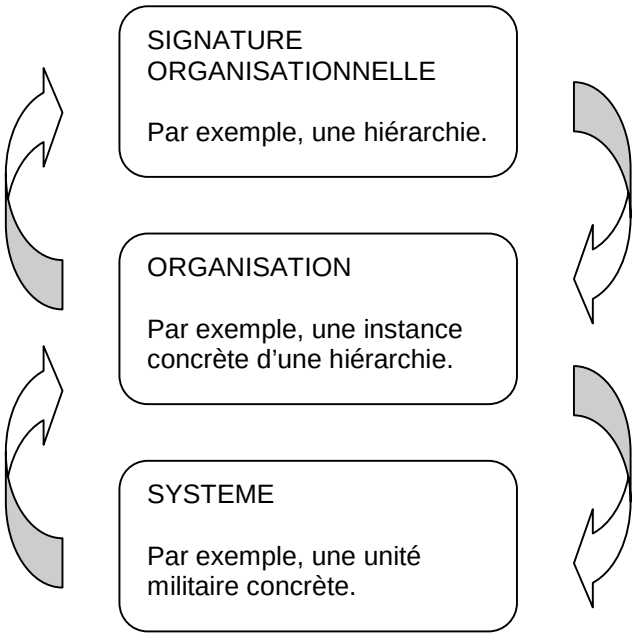


Figure 3 : Les niveaux différents d'organisation.

SMA Minimaux	**Ferber**	**Maturana**
système	système	organisme
structure / organisation	*organisation concrète*	*structure*
signature organisationnelle	structure organisationnelle / structure	organisation

Tableau 2 : Comparaison de terminologie.

Dans le domaine des SMA, les notions de *rôle* et de *tâche* sont directement liées à celle d'organisation. En essence, l'organisation est vue comme un mécanisme pour gérer la distribution de **rôles** et/ou de **tâches** sur les différents agents, afin de garantir une **fonctionnalité globale**. Par exemple, la notion de rôle est essentielle dans les méthodologies récemment développées telles que Cassiopée [Collinot 96] ou Aalaadin [Gutknecht 98].

La manipulation de rôles exige des capacités cognitives, impliquant que *soit* l'organisation soit déterminée par le concepteur, *soit* qu'elle soit déterminée par des agents cognitifs. Si on veut construire des systèmes dans lesquels les agents mêmes changent leur organisation, on est obligé d'utiliser des agents cognitifs. Si on veut utiliser uniquement des agents réactifs, on est obligé d'imposer une organisation statique sur les agents. Notons que - en général - il est très difficile de réaliser la traduction automatique, dans les deux sens, entre les rôles et les propriétés des agents au niveau réactif. Il s'agit d'une instance spécifique du *Symbol Grounding Problem*.

Puisque nous voulons étudier la dynamique organisationnelle non seulement pour les SMA composés d'agents cognitifs, mais également pour des systèmes composés uniquement d'agents réactifs, nous avons évité la notion de rôle et de fonctionnalité explicite (symbolique). Donc, nous avons basé notre concept d'organisation sur des relations de groupement. Le groupement (alias organisation alias structure) détermine l'espace 'social' des

agents, restreignant les interactions entre agents. Dans les limites imposées par cette organisation, les agents interagissent, et en interagissant ils changent leur organisation. Ce processus constitue la dynamique organisationnelle que nous étudions.

Soulignons que notre notion d'organisation n'est pas incompatible avec la notion classique dans le domaine. Au contraire, comme nous le verrons dans le chapitre II.5, chaque organisation « classique » implique automatiquement des relations de groupement. Nous étudions le concept d'organisation au niveau le plus bas, qu'on retrouve, implicitement ou explicitement, dans tout Système Multi-Agents. Par exemple, notre modèle permet l'étude théorique de la migration des agents (de groupe à groupe) qu'on retrouve dans tout SMA ayant une organisation dynamique.

Dans cette thèse, nous proposons un modèle d'un type de systèmes, plutôt qu'un modèle d'organisation en soi. Notre modèle décrit une classe très spécifique de SMA. En effet, le comportement à long terme de tous les Systèmes Multi-Agents Minimaux est explicitement défini. Ce choix nous a permis de limiter les variables libres à un minimum, facilitant l'étude de la dynamique organisationnelle. Par contre, il est moins adapté pour construire des systèmes généraux. Toutefois, il s'avère que les Systèmes Multi-Agents Minimaux, malgré leur comportement fixe, s'adaptent facilement à un nombre important de problèmes. Le comportement à long terme que nous avons choisi semble être d'une nature relativement universelle.

I.1.7 Le maintien de l'intégrité fonctionnelle

Une partie importante des recherches de cette thèse a été effectuée dans le cadre du marché CTI N96 1B 06 entre l'INPG, l'Institut National Polytechnique de Grenoble, et le CNET, le Centre de Recherche et de Développement de France Télécom [CNET]. Le Cahier des Clauses Techniques Particulières de ce marché spécifie :

> « Le but de ce présent marché est d'apporter des éléments de solution au problème du maintien de l'intégrité fonctionnelle face à la dynamique des organisations dans les Systèmes Multi-Agents Ouverts (SMAO). Un SMAO peut être défini comme un système multi-agents dans lequel des agents peuvent librement décider d'entrer ou de sortir, de migrer de sites d'exécution, de faire évoluer leur offre de services, d'interagir avec d'autres agents. »

La recherche du marché a été prévue sur 36 mois et a commencé en mars 96. Les progrès sont documentés dans six rapports semestriels [CTI 96-99]. Contrairement à cette thèse, ces rapports ne sont pas dans le domaine public.

Le Cahier des Clauses Techniques Particulières du marché spécifie un certain nombre de tâches, considérées comme importantes dans la réalisation du but. Dans cette spécification, la nature de certaines de ces tâches suggérait l'utilisation d'agents cognitifs. Pour des raisons que nous avons indiquées ci-dessus, les travaux de notre thèse se situent à un niveau non cognitif et peuvent être appliqués à des systèmes composés d'agents réactifs et/ou des agents cognitifs. Les travaux de Jaroslav Kozlak et d'Alexandre Ribeiro, contribuant également (et partiellement) au marché, sont plus ciblés sur les agents cognitifs. Ces travaux seront publiés dans deux thèses de doctorat, à paraître au cours de l'année 1999.

La notion de Systèmes Multi-Agents Ouverts est directement liée à celle de systèmes ouverts informatiques en général (*Open Information Systems*). Carl Hewitt, un des chercheurs très actifs dans ce domaine, les définit

comme des systèmes « which are always subject to unanticipated outcomes in their operation and which can receive new information from outside themselves at any time. » L'importance croissante de grands systèmes ouverts, tel que l'Internet, a stimulé considérablement leur étude. Toutefois, notons que tout système utile est nécessairement ouvert à un certain degré. En effet, un système complètement clos ne peut pas interagir avec des utilisateurs et ne peut pas réaliser une fonctionnalité voulue.

Par contre, la nature ouverte d'un SMAO peut compromettre son intégrité fonctionnelle. Par exemple, si tous les agents émigrent du système, sa fonctionnalité est nécessairement réduite à zéro. La fonctionnalité est un concept très utilisé dans notre société, mais toutefois délicat à définir. Prenons un système informatique décomposant un nombre en nombres premiers. Sa fonctionnalité n'est pas déterminée seulement par sa fonction mathématique. Par exemple, si le système prend quelques jours pour finir une certaine décomposition, il peut être considéré fonctionnel par des mathématiciens, mais non fonctionnel par des agents secrets essayant de craquer rapidement un code. La nature des utilisateurs, jouant un rôle essentiel en déterminant la fonctionnalité d'un système, est généralement difficile à formaliser. Même le temps de réponse réel d'un système ne correspond par directement au temps de réponse perçu.

D'une manière empirique, nous pouvons définir la fonctionnalité d'un système en fonction se sa *réutilisation* par les utilisateurs : un utilisateur réutilisera plus facilement un système fonctionnel qu'un système non fonctionnel. Cette définition peut être utile dans certains contextes, par exemple lorsqu'on construit des programmes évolutifs sur l'Internet. Dans cet exemple, les programmes peuvent être vus comme des algorithmes évolutifs ayant leur réutilisation comme mesure de fonctionnalité et donc comme fonction de *fitness* [Holland 75]. Toutefois, dans d'autres contextes, nous souhaitons une définition moins empirique.

Dans les modèles théoriques des SMA ouverts, la notion de fonctionnalité est souvent directement liée à celle de rôle. Par exemple, les travaux de Antônio Carlos da Rocha Costa sont basés sur la notion de *rôle fonctionnel* [Rocha Costa 93] [Rocha Costa 94] [Rocha Costa 96]. Dans notre modèle, nous ne pouvons pas utiliser cette assimilation entre fonctionnalité et rôle puisque nous voulons - pour des raisons indiquées ci-dessus - éviter la manipulation explicite de rôles.

Des écoles dans l'Intelligence Artificielle plus proche de la Vie Artificielle proposons la notion de *fonctionnalité émergente*, évitant la description symbolique de fonctions ou de rôles au sein système. Luc Steels nous donne la définition suivante [Steels 94] :

> « A behavior is emergent if it can only be defined using descriptive categories which are not necessary to describe the behavior of the constituent components. An emergent behavior leads to emergent functionality if the behavior contributes to the system's self-preservation and if the system can build further upon it. »

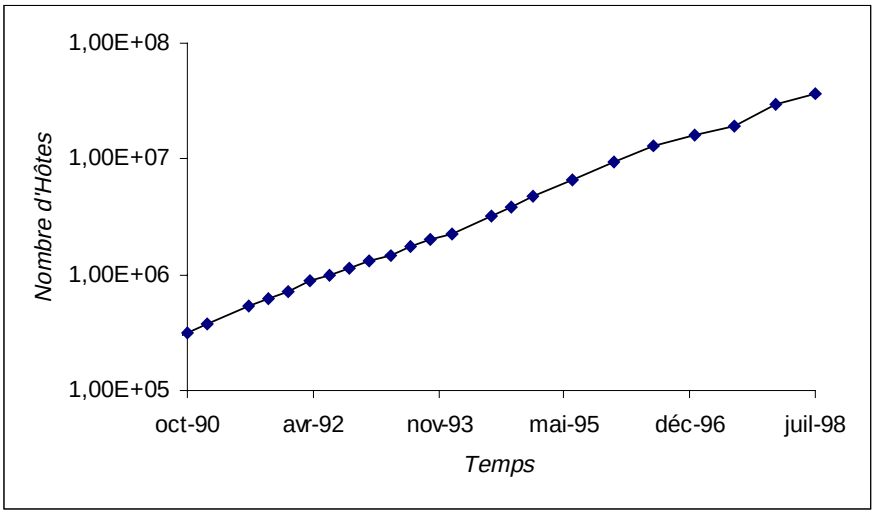
Dans cette définition, la fonctionnalité d'un système est liée au *maintien de son état dans certaines limites*. Dans ce cas, le maintien de l'intégrité fonctionnelle correspond au maintien de certains invariants du système. Dans cette thèse, nous prenons ce point de vue plutôt qu'une perspective basée sur la notion de rôle. Nous nous concentrons sur la structure (organisation) de systèmes et sur son évolution. L'étude du maintien de l'intégrité fonctionnelle même ne se situe pas au centre de cette thèse, mais fait partie de son contexte.

Un SMA ouvert est nécessairement un SMA ayant une organisation dynamique : la migration d'agents implique des changements dans l'organisation. Donc, l'étude de la dynamique organisationnelle de ces systèmes constitue une partie essentielle de l'étude du maintien de l'intégrité fonctionnelle. Le modèle des Systèmes Multi-Agents Minimaux, sujet de cette thèse, est un outil facilitant cette étude. Ceci constitue notre contribution principale à l'étude du maintien de l'intégrité fonctionnelle.

I.1.8 L'Internet

Comme nous l'avons indiqué au début de ce chapitre, ces dernières décennies ont été caractérisées par une croissance exponentielle de la complexité de certains systèmes artificiels. Au niveau de l'ordinateur isolé, ce phénomène peut être observé dans le domaine des processeurs et des logiciels. Plus récemment, le même phénomène se manifeste - d'une manière spectaculaire - à un niveau plus macroscopique : l'Internet, le réseau à l'origine conçu comme un projet DARPA [Gaffin 94], a explosé et compte actuellement plusieurs dizaines de millions d'ordinateurs.

Le graphique 2 montre l'évolution du nombre d'hôtes sur l'Internet entre octobre 1990 et juillet 1998 [Wizards]. Notons que les données à partir de janvier 1998 sont mesurées d'une manière plus précise, et que les données entre janvier 1995 et juillet 1997 sont ajustées pour éviter une discontinuité dans la courbe.



Graphique 2 : Evolution du nombre d'hôtes sur l'Internet.

Le World Wide Web

Le développement exponentiel de l'Internet doit surtout être attribué à l'intérêt croissant que le grand public montre pour ce réseau. Plus que les autres services sur l'Internet, le World Wide Web (Web), facile à utiliser, d'une nature multimédia et extrêmement ouvert, a séduit le grand publique de « se brancher ». Une fois connecté, il découvre rapidement l'intérêt des autres services, tels que le courrier électronique, les forums et les salons de conversation.

La nature associative du Web, cette toile mondiale inventée par Tim Berners-Lee il y dix ans [Berners-Lee 90] [Berners-Lee 92], constitue une partie importante de son charme et de sa facilité d'utilisation. En suivant les liens de page en page, les utilisateurs suivent un parcours reflétant à la fois leurs intérêts et les intérêts des

concepteurs des pages. Cette façon de naviguer n'est pas seulement naturelle, mais peut mener à la découverte imprévue d'informations importantes. Le phénomène de *serendipidity* (trouver par hasard, dans une recherche, un nouvel élément plus important que l'élément recherché) joue un rôle central dans l'utilisation du Web. Notons que l'idée d'utiliser une hyper-structure ne date pas de cette décennie. Ted Nelson a introduit la notion de hyper-texte déjà dans les années 60 [Nelson 67]. Cette mode de recherche peut être très utile lorsque l'on n'a pas d'idée précise de ce que l'on veut. Par contre, si on souhaite effectuer une recherche systématique sur un sujet bien défini, la nature associative du Web peut s'avérer catastrophique. Comme nous le verrons dans le chapitre IV.2, la recherche d'informations sur le Web constitue un sujet de recherche (académique et commerciale) très important.

Il est peu probable que le Web aurait connu sa croissance impressionnante, s'il était resté un système en mode hyper-texte. Son contenu multimédia a sans doute contribué à sa popularité. D'ailleurs, l'évolution vers une nature de plus en plus multimédia est une des caractéristiques importantes du Web. Le moindre surplus de bande passante est exploité pour publier et télécharger des images, des sons, de la musique, de la vidéo (statique et *streaming*). De plus en plus, le Web devient un spectacle audiovisuel concurrençant les médias traditionnels.

Toutefois, une différence essentielle entre les médias traditionnels et le Web est que les premiers sont fermés et le deuxième est ouvert. En effet, dans les médias classiques, peu de gens publient (et peuvent publier). Par contre, sur le Web, tout le monde peut publier (et beaucoup le font). Jamais dans l'histoire de l'homme, un médium de publication n'a été tellement démocratique.

Deux mondes

En raison de sa nature ouverte, le Web est devenu un dépôt énorme d'*éléments culturels*. Nous utilisons le terme « élément culturel » pour indiquer qu'il ne s'agit pas d'*informations* dans le sens classique du mot. En effet, sur le Web, n'importe qui peut publier n'importe quoi, car le coût de publication est relativement modeste. La publication de données n'est pas nécessairement soumise à un processus de vérification et de « reviewing ». Toutefois, les éléments publiés ont une certaine valeur, même s'ils sont - d'un point de vue objectif - complètement faux. En effet, même s'ils ne sont appréciés que par quelques personnes, ils font partie de notre *culture humaine*.

Le Web n'est pas seulement un miroir électronique de la culture humaine, mais des parties importantes de notre culture n'existent que sur le Web (comme un nombre croissant de documents Web). Une partie de notre culture quotidienne est en train de migrer vers ce monde électronique. Les autres services de l'Internet, tels que les forums, renforcent ce processus. De plus en plus, l'homme interagit sur l'Internet, en échangeant des messages électroniques.

Dans notre vision, l'Internet peut jouer un rôle clé dans l'avenir de l'Intelligence Artificielle. En effet, nous sommes convaincus que l'intelligence est un phénomène socioculturel. Le Test de Turing (TT) a pour nous non seulement une valeur historique, mais également une valeur scientifique. Le TT, légèrement modifié pour le rendre plus valide statistiquement, nous semble le seul moyen pour déterminer si un système est artificiellement intelligent. Autrement dit, notre définition de l'IA est basée sur le TT. A l'heure actuelle, la réalisation de systèmes capables de passer le TT est hors de notre portée, mais ce fait ne nous semble pas être un argument valide pour modifier notre définition.

Considérant notre vision de l'Intelligence Artificielle, l'Internet est d'une importance cruciale, car elle rend une partie de la culture humaine directement accessible à des entités électroniques, tels que les agents artificiels. Nous croyons que - en interagissant dans cet espace commun - agents et humains peuvent développer un langage commun et s'approcher les uns des autres [Van Aeken 96b]. Une fois une culture commune établie, la perspective de voir des entités artificielles passer le TT deviendra plus réaliste.

Evidemment, nous sommes encore très éloignés de cette situation. A l'heure actuelle, peu d'agents autonomes existent sur l'Internet. Il n'y a pas encore une technologie normalisée supportant la mobilité des agents. Toutefois, pour garantir leur survie, les agents autonomes doivent être capables de migrer de site à site. De plus, pour qu'ils puissent construire une culture et partager des connaissances, ils doivent s'organiser comme des communautés. A l'heure actuelle, il n'existe pas encore de telles populations sur l'Internet. Finalement, pour que les deux cultures puissent s'approcher, il doit exister une véritable interaction entre l'homme et l'agent. Ce sujet est encore peu étudié. Toutefois, dans notre vision, il est d'une importance essentielle.

L'interaction entre l'homme et l'agent pose le problème de l'enracinement des symboles. Puisque nous vivons dans deux mondes différents, dans quel monde devons nous enraciner nos symboles ? C'est à ce niveau que nous espérons contribuer à la recherche de l'intelligence artificielle. Dans cette thèse, nous étudions l'organisation dynamique d'agents à un niveau sous-cognitif, le niveau du *groupement*. Ce niveau existe, implicitement ou explicitement, dans tout ensemble d'agents, naturels ou artificiels. Appartenir ou non à un groupe est une notion universelle, qui peut être la base de toute communication entre entités différentes. Dans cette thèse, nous proposons des outils pour étudier, d'une manière minimale, mais exacte, cette notion. Nous croyons qu'une meilleure compréhension de la notion peut mener à des mécanismes pour enraciner les symboles échangés entre hommes et agents. Toutefois, la construction de tels mécanismes dépasse les limites de cette thèse.

Dans la partie suivante, nous introduirons le modèle de Systèmes Multi-Agents Minimaux. Ce modèle est à la base de notre étude de la dynamique organisationnelle dans des systèmes composés de plusieurs agents.

CHAPITRE II.1

MOTIVATIONS

La perfection est atteinte non quand il ne reste rien à ajouter,
mais quand il ne reste rien à enlever.
Antoine de Saint-Exupery

II.1.1 Un modèle minimal

Dans la première partie de cette thèse, nous avons présenté un aperçu du contexte de nos travaux. Ils se situent dans l'Intelligence Artificielle, cette science très particulière et très liée à l'informatique. Notre ambition est de contribuer à cette science en proposant un modèle très simple adapté à l'étude de la dynamique organisationnelle des Systèmes Multi-Agents.

Il y a une certaine ironie dans notre approche. D'un côté se trouve le phénomène de l'intelligence, dont la complexité, bien sous-estimée par les pionniers de l'IA, est profondément affirmée aujourd'hui. Sa problématique est tellement dure que, même après toutes ces années, nous n'avons pas encore établi une méthodologie généralement acceptée dans le domaine. D'un autre côté se trouve notre modèle qui - comme nous le verrons dans cette partie - est extrêmement simple. Souhaitons-nous nous attaquer à un des phénomènes observés les plus complexes avec un des plus simples modèles imaginables ? En effet. Nous voulons réduire la complexité du phénomène en en extrayant un élément de base que nous pouvons étudier isolément.

Evidemment, la question est de savoir si nous avons isolé un élément de base ou juste un détail. La validation du modèle, soit par prédiction de phénomènes, soit par construction de systèmes utiles, sert exactement à répondre à cette question. Pourtant, hors de cette validation, nous pouvons, dans ce chapitre introductif, motiver nos choix d'une manière intuitive, se fiant à notre *sens commun*. Ceci peut aider à la compréhension du modèle et de notre approche méthodologique. Il est bien évident que, en aucun cas, ces motivations ne remplacent la validation objective.

Soulignons que notre modèle est un modèle informatique, ne modélisant pas directement un phénomène naturel tel que l'intelligence. Il s'agit d'un modèle des systèmes informatiques à un niveau abstrait. Notre modèle se situe dans la même classe que les machines Turing, les réseaux neuronaux ou encore les automates cellulaires. Evidemment, le modèle peut être utilisé pour construire des modèles du naturel. Inversement, nous nous sommes inspirés des modèles naturels dans la construction de notre modèle.

Etant chercheur dans le domaine des Systèmes Multi-Agents (SMA), nous sommes convaincus de l'importance de la dimension sociale et culturelle de l'intelligence. Nous croyons que des phénomènes tels que l'intelligence sont directement liés aux interactions entre plusieurs agents. Donc, nous avons essayé de construire un modèle minimal de systèmes composés de plusieurs agents, i.e. de plusieurs processus autonomes. Assez logiquement, nous avons baptisé ce modèle les **Systèmes Multi-Agents Minimaux** ou **SMAM**.

Dans notre modèle, nous faisons abstraction de la nature des agents qui peut être cognitive, réactive ou hybride. Ce qui nous intéresse, c'est de modéliser comment les agents peuvent interagir entre eux. En effet, lorsqu'on rassemble plusieurs agents, on crée - implicitement ou explicitement - un espace doté d'une certaine topologie. Cette topologie détermine la facilité avec laquelle deux agents du système peuvent interagir : les agents proches dans l'espace peuvent interagir plus facilement que les agents lointains. Il s'agit de la notion d'**organisation** à un niveau fondamental, à savoir le niveau du **groupement** des agents. Toutes les autres interactions dans le système, celles sur les niveaux les plus élevés inclus, s'appuient sur cette organisation de base. Notons qu'il ne s'agit pas d'un espace externe dans lequel les agents sont placés, mais de l'espace « social » qui émerge automatiquement une fois qu'on rassemble plusieurs agents.

Lorsqu'on rassemble deux agents, on crée - par la définition même du concept « rassembler » - le **couple** des deux, constituant une troisième entité, *que nous considérons comme un agent aussi*. Donc, lorsqu'on ajoute encore un agent, un deuxième couple est formé composé du couple existant et l'agent ajouté. La formalisation de ce processus est à la base de notre théorie des Systèmes Multi-Agents Minimaux.

Notre approche implique que les agents sont organisés d'une manière binaire. Elle ne permet pas l'existence d'ensembles mathématiques de taille supérieure à deux, i.e. des groupes non structurés composés de plus de deux agents : le couple est le seul degré de liberté. Notons toutefois que - du point de vue de l'observateur - les agents peuvent sembler constituer un ensemble non structuré si leur organisation change arbitrairement et rapidement dans le temps.

Cette organisation obligatoirement binaire des agents ne nous gène pas. Au contraire, elle limite l'ambiguïté spatiale des agents. Elle constitue une restriction qui nous permet d'effectuer des analyses très strictes et très détaillées. De plus, elle exprime une intuition très importante sur les espaces, naturels ou artificiels. Par exemple, dans l'espace physique quotidienne, les objets sont structurés d'une manière très stricte (binaire, si on veut) et leurs relations spatiales sont déterminées sans ambiguïté. En effet, c'est exactement cette structuration stricte qui nous permet de mesurer l'espace. D'une manière similaire, les données dans la mémoire d'un ordinateur sont structurées d'une façon binaire. A tout moment, l'ordinateur physique impose une structure sur les éléments qu'il manipule. Donc, notre minimalisme binaire, qui nous convient tellement d'un point de vue analytique, est conforme à une intuition que nous partageons tous.

Soulignons que les agents sont des entités actives. Donc, l'organisation d'un système composé de plusieurs agents est typiquement très dynamique : les agents migrent dans l'espace d'une manière continue. En effet, l'intérêt d'un grand nombre de systèmes n'est pas directement dans leur organisation, mais dans leur dynamique organisationnelle. Une définition complète de notre modèle doit inclure la définition de la nature des actions des agents. Nous avons choisi une définition qui est d'une simplicité maximale, mais conforme à nos intuitions, liant notre modèle à la Systémique, la science étudiant les principes universels d'organisation.

II.1.2 Un modèle systémique

Ludwig von Bertalanffy était le premier à proposer, sous le nom de « General Systems Theory » le concept de la Systémique [von Bertalanffy 68]. Au cœur de ce concept, nous avons l'hypothèse que nous pouvons trouver des principes universels d'organisation applicables dans tous les domaines scientifiques. Francis Heylighen de la Vrije Universiteit Brussel formule la notion comme suit [Principia Cyber.] :

> « General Systems Theory is based on the assumption that there are universal principles of organization, which hold for all systems, be they physical, chemical, biological, mental or social. The mechanistic world view seeks universality by reducing everything to its material constituents. The systemic world view, on the contrary, seeks universality by ignoring the concrete material out of which systems are made, so that their abstract organization comes into focus. »

Dans la Systémique, l'importance d'un concept est, par définition, liée au degré dont le concept peut être appliqué dans un grand nombre de domaines scientifiques différents. En ce qui concerne la dynamique organisationnelle des systèmes, la notion d'**entropie** et sa **maximisation** joue un rôle important dans la Systémique. En effet, le concept d'entropie est manipulé dans un grand nombre de disciplines. Par exemple, on le retrouve dans les sciences physiques, dans la théorie de l'information et du codage, dans le domaine des systèmes dynamiques, dans la logique et la théorie des algorithmes, dans l'inférence statistique et la prédiction, dans les sciences économiques, dans la biologie et dans les sciences sociales.

A cause de la nature différente de ces domaines, la définition de l'entropie est différente de domaine à domaine et le lien entre ces types différents d'entropie est souvent très dur à démontrer. Toutefois, la notion d'entropie et sa maximisation expriment, dans tous ces domaines, toujours les mêmes intuitions. En particulier, l'évolution de systèmes vers un état d'entropie maximal exprime l'intuition que ces systèmes évoluent, d'une manière irréversible, d'un état moins équilibré vers un état plus équilibré. Prenons l'exemple classique de deux pièces séparées par une porte. L'air dans la première pièce est chaud. L'air dans la deuxième pièce est froid. Lorsqu'on ouvre la porte entre les deux pièces, l'air se mélange - d'une manière très complexe - jusqu'à il soit tiède dans les deux pièces. Donc, ce système particulier évolue de l'état (1, 0) vers l'état (0.5, 0.5), étant d'un équilibre maximal.

Guidés surtout par l'intuition, nous avons défini l'évolution naturelle d'un SMAM comme la maximisation de son équilibre. L'équilibre d'un SMAM est défini en termes de son entropie, dont le calcul est inspiré de la formule classique d'entropie des sciences de l'information. Il nous semble difficile de trouver une autre définition de l'évolution naturelle de systèmes composés de plusieurs agents qui soit à la fois très simple - permettant des analyses quantitatives - et conforme aux intuitions.

Notre modèle exprime, jusqu'à un certain degré, l'intuition sous-jacente à la notion d'entropie étudiée dans la Systémique. Le modèle propose également une définition minimale de structure, i.e. d'organisation fondamentale, indépendante de la nature spécifique des agents constituant le système. Donc, on peut se poser la question de savoir si notre modèle peut jouer un rôle dans la Systémique. A cause de sa nature minimale et le fait qu'il exprime des intuitions importantes, nous croyons que c'est le cas. En effet, dans son rôle de meta-science, la Systémique essaie d'extraire des facteurs communs entre un grand nombre de modèles différents. Les facteurs communs sont nécessairement simples, mais doivent exprimer des notions universelles.

Le potentiel de la Systémique est important, car elle ouvre les portes vers l'interdisciplinarité. Elle permet aux domaines différents de s'ajuster et de partager les notions communes. Ceci nous permettra, en théorie, d'avancer dans nos sciences d'une manière plus économique et plus rapide. Pour ces raisons, la Systémique nous est très chère. Toutefois, malgré notre fascination pour cette science, nous ne sommes pas des chercheurs en Systémique, mais en Informatique. Si nous voulons évaluer notre modèle dans le contexte de la Systémique, nous sommes obligés de le faire en coopération avec des spécialistes du domaine. Dans le cadre de cette thèse, nous toucherons à la Systémique vers la fin de cette partie, mais cette excursion restera superficielle.

II.1.3 Un modèle sans connotations

Les avantages d'un modèle minimal sont assez clairs du point de vue du systématicien. Toutefois, dans des domaines tel que l'Intelligence Artificielle ou l'informatique en général, les systèmes minimaux sont plutôt une exception qu'une règle.

Lorsqu'on essaie de modéliser un système complexe, il est compréhensible qu'on essaie d'incorporer, dans le modèle, un maximum de caractéristiques du système. Ceci peut être réalisé d'une manière explicite, en codant les caractéristiques en dur, ou d'une manière implicite, en les codant comme des phénomènes émergents.

Dans la première partie de cette thèse, nous avons détaillé la différence entre ces deux approches. Illustrons la encore une fois à l'aide d'un exemple. Prenons le cas d'un psychologue qui observe, dans un agent naturel, l'existence de croyances, désirs et intentions. Partant de cette observation, le psychologue peut construire un modèle informatique dans lequel les agents manipulent des variables nommées « croyances », « désirs » et « intentions ». Il s'agit d'une approche tout à fait valable, *pourvu* que le phénomène observé reflète un mécanisme de base et non son interprétation plus ou moins arbitraire de la part de l'observateur. L'approche alternative est de trouver des mécanismes de base situés à un niveau inférieur au niveau observé. Une fois ces mécanismes implémentés, les phénomènes que l'on veut reproduire émergent au niveau de l'observateur.

La reconnaissance du rôle de l'observateur dans la deuxième approche nous semble très importante. En effet, la première approche lie le modèle construit directement à l'observateur et incorpore toutes ses connotations personnelles liées aux concepts implémentés. Par exemple, un psychologue, un sociologue et un épistémologue peuvent avoir des connotations très différentes du concept de « croyance ». En conséquence, leurs modèles informatiques, construits selon la première approche, sont différents aussi et n'ont pas nécessairement de sens pour les autres scientifiques. Par contre, un seul modèle construit selon la deuxième approche peut reproduire tous les phénomènes observés, respectivement, par le psychologue, le sociologue et l'épistémologue. De plus, puisque le modèle se situe à un niveau inférieur, il peut plus facilement être partagé par des scientifiques dans des disciplines différentes.

Un modèle développé au sein d'une discipline spécifique, en utilisant des notions propres à cette discipline, peut être vu comme la projection d'un phénomène dans l'espace « scientifique » associé au domaine. Un seul phénomène mènera à des projections différentes dans des domaines différents. Puisque les espaces associés aux disciplines différentes ne sont pas identiques, les projections ne peuvent pas directement être échangées entre scientifiques. Ceci est illustré par la figure 1. Par contre, un modèle construit dans un espace plus général, i.e. comptant plus de dimensions, peut plus facilement être partagé par des scientifiques différents. Puis, les modèles spécifiques aux disciplines peuvent être construits par projection. Cette approche est illustré par la figure 2.

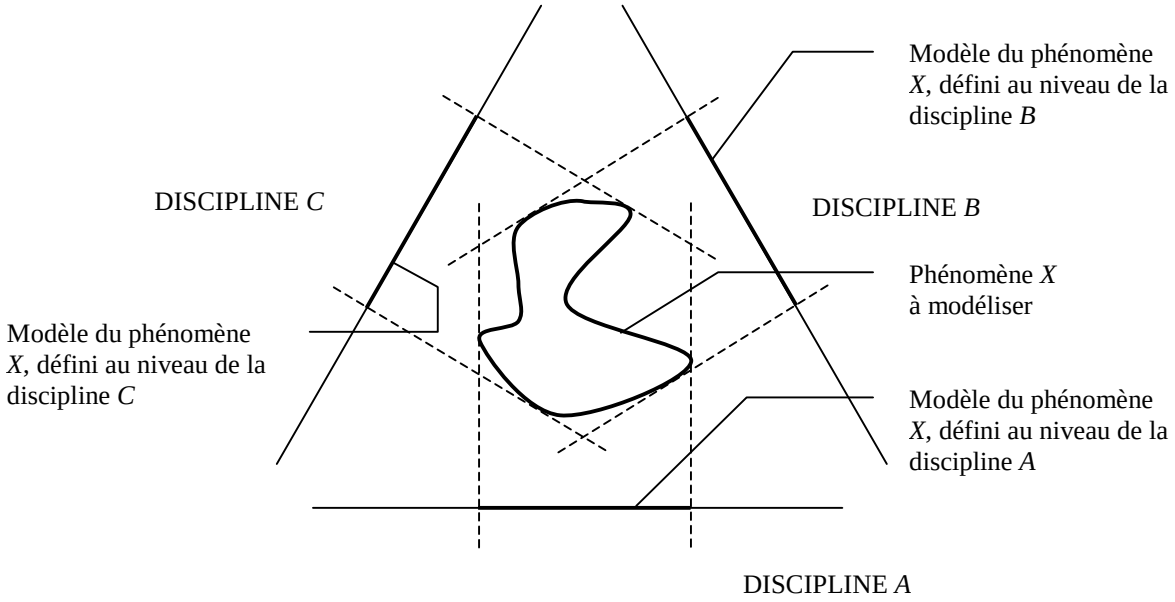


Figure 1 : Modèle défini au niveau des différentes disciplines.

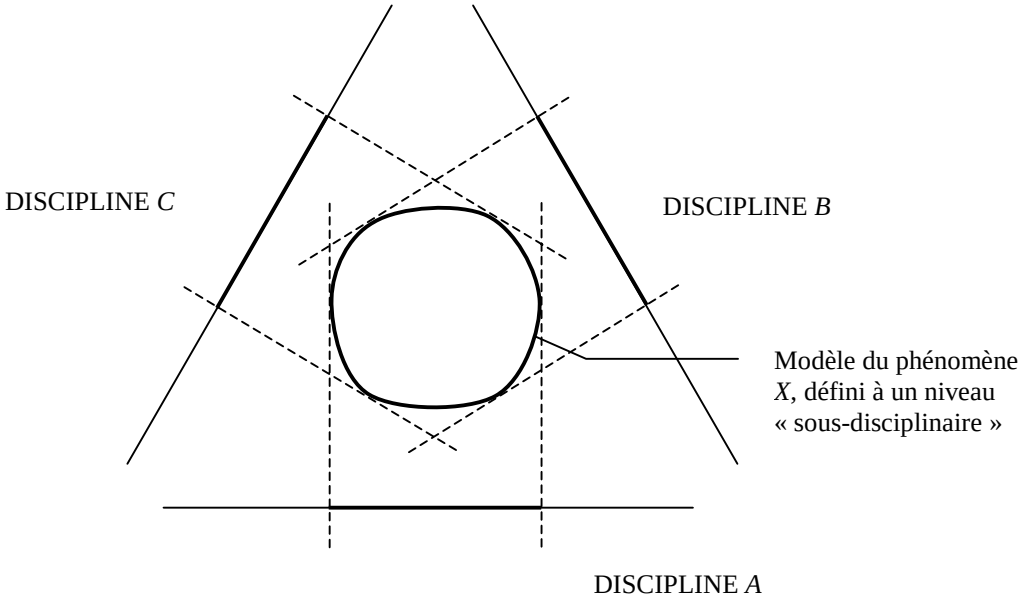


Figure 2 : Modèle défini à un niveau « sous-disciplinaire ».

Dans notre thèse, nous avons poursuivi la deuxième approche à un degré maximal. En effet, le modèle des SMAM se situe à un niveau tellement bas que les connotations liées aux concepts manipulés sont réduites à un minimum. La notion des SMAM est facile à formaliser sans ambiguïté. De plus, les SMAM et leur évolution peuvent être mesurés d'une manière quantitative.

L'absence relative de connotations dans notre modèle non seulement limite des implémentations ambiguës, mais rend le modèle plus accessible à un grand nombre de chercheurs, ou même de non chercheurs.

Evidemment, il n'y pas que des avantages à notre approche minimale. C'est surtout d'eux que nous parlons dans cette section, puisqu'elle est dédiée à nos motivations. Toutefois, nous sommes conscients que notre approche implique au moins deux inconvénients importants.

Le premier est lié au niveau très bas auquel se situe notre modèle. En effet, d'un certain point de vue, il ne se situe pas au niveau de l'Intelligence Artificielle, ni au niveau plus bas de la Vie Artificielle, mais au niveau de ce qu'on peut appeler la Thermodynamique Artificielle. En effet, nous ne définissons que des simples structures actives qui évoluent selon des lois simples et rigides. Evidemment, à partir de ce modèle de base, nous pouvons construire des modèles plus sophistiqués. Comme alternative, nous pouvons appliquer notre modèle directement à des niveaux plus élevés. Dans ce cas, notre modèle est plus abstrait relativement au sujet, mais toujours utile dans l'analyse de la dynamique organisationnelle de systèmes composés de plusieurs agents.

Le deuxième inconvénient est lié à la généralité du modèle. Parce qu'il est tellement générique, il ressemble superficiellement à un grand nombre d'autres modèles. En particulier, il peut invoquer l'image des arbres binaires, même si la correspondance n'est que superficielle. Ceci s'est avéré un problème chronique dans nos efforts à communiquer notre modèle à la communauté scientifique. Donc, ironiquement, notre souhait d'éviter des connotations aux niveaux élevés peut introduire des malentendus, peu nombreux mais toujours gênants, aux niveaux les plus bas.

Les deux inconvénients que nous avons identifiés ne sont pas sans importance. Toutefois, il nous semble que les avantages de notre approche surpassent les inconvénients. Soulignons, cependant, que seule la validation scientifique peut établir la valeur objective de notre modèle. Pour l'instant, la validation est effectuée surtout par la réalisation d'applications concrètes. Nous présenterons ces aspects de nos travaux à partir de la troisième partie de cette thèse. Dans un premier temps, étudions - dans les chapitres suivants - le modèle même.

CHAPITRE II.2

DEFINITION DES SYSTEMES MULTI-AGENTS MINIMAUX

Make things as simple as possible, but not simpler.
Albert Einstein

II.2.1 Agents atomiques et agents composés

Qu'est-ce qui différencie les Systèmes Multi-Agents d'autres systèmes ? Quelles propriétés sont **nécessaires** et **suffisantes** pour définir cette classe de systèmes ? Quel est le modèle le plus simple imaginable qui formalise l'essentiel des Systèmes Multi-Agents ? Telles sont les questions qui nous ont motivés pour développer le modèle des Systèmes Multi-Agents Minimaux et qui sont à la base de sa définition.

Une définition récursive

Le concept des « Systèmes Multi-Agents » combine la notion de « système » avec les notions de « multi » et d'« agent ». La sémantique du préfix « multi » est simple : elle indique que le système est typiquement **composé de deux ou plusieurs agents (1)**. La signification du mot « agent » est, comme nous l'avons vu dans une section antérieure, moins évidente. Toutefois, en moyennant les définitions du concept d'agent les plus utilisées, nous définissons un agent comme **un processus**, i.e. quelque chose qui agit, **doté d'une certaine autonomie (2)**, i.e. qui n'est pas complètement contrôlé par un autre processus.

Essayons d'être encore plus exacts. Afin de distinguer la classe de systèmes très spécifique que nous définissons dans ce chapitre de la classe générique des SMA, nous l'appelons la classe **des Systèmes Multi-Agents Minimaux**, ou **SMAM**. Formulons la propriété (1) plus exactement en introduisant la définition suivante :

Un SMAM est *soit* un agent atomique, *soit* un agent composé de deux SMAM.

Comme l'indique son nom et comme le suggère la définition, un agent atomique ne peut pas être plus décomposé. La définition est récursive et basée sur une décomposition en deux parties. Le choix du nombre deux est logique, vue notre tentative de trouver un modèle minimal.

Un SMAM est toujours un agent, donc - selon la définition (2) - un processus doté d'une certaine autonomie. Si nous voulons définir le concept des SMAM sans laisser d'ambiguïtés, nous devons définir également le comportement de ces agents. A cause de la définition récursive des SMAM, ce comportement doit être définissable indépendamment de la taille ou de la structure des agents. Nous définirons plus tard dans ce chapitre le comportement choisi. Dans un premier temps, étudions de plus près la structure des SMAM. A cette fin, introduisons une première représentation graphique basée sur le concept des arbres binaires. Nous l'appelons la **représentation arborescente** des SMAM. En utilisant cette représentation, un agent atomique se présente comme suit :

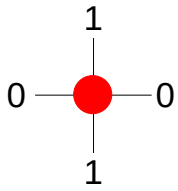


Figure 1 : L'agent atomique.

Le disque rouge (gris en reproduction monochrome) représente l'agent atomique. Le segment horizontal indique le niveau de l'agent dans la structure du SMAM complet. Comme le SMAM ne compte qu'un agent, l'agent se trouve au niveau maximal, à savoir le niveau zéro. Le segment vertical identifie le rang de l'agent atomique. Comme le SMAM ne compte qu'un agent atomique, il porte le rang un. Le but principal des segments horizontaux et verticaux et de leurs étiquettes est de pouvoir connaître rapidement les dimensions du SMAM et de pouvoir identifier facilement ses agents atomiques.

Lorsqu'on ajoute un agent atomique au SMAM de la figure 1, **il se forme- par définition - l'agent composé** de l'agent original et de l'agent ajouté ! Dans le monde des SMAM, **1 + 1 = 3** ! L'agent résultant est illustré dans la figure 2 :

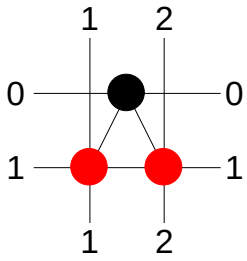


Figure 2 : L'agent composé.

Un agent composé est visualisé comme un disque noir. Notons que **l'agent composé représente le couple** des deux agents atomiques. Le danger d'une représentation basée sur les arbres binaires est que la structure d'un SMAM peut être interprétée comme un ensemble de relations hiérarchiques entre des agents autrement non liés. Ceci n'est pas exact, car l'agent composé de la figure 2 n'est pas indépendant des deux agents atomiques : il ne peut pas exister sans eux, car il représente leur *couple*. Comme nous le verrons plus loin, le comportement des agents est également basé sur le concept du couple. S'il existe un concept singulier au centre de la théorie des SMAM, c'est sans doute celui du **couple**.

Nous avons développé une deuxième représentation graphique qui est moins facile à manipuler d'un point de vue analytique, mais qui représente plus fidèlement les relations entre les agents d'un SMAM. Nous l'avons baptisé la **représentation triangulaire** des SMAM. La figure 3 montre les SMAM des figures 1 et 2, en utilisant cette représentation.



Figure 3 : Représentation triangulaire des SMAM des figures 1 et 2.

Dans la représentation triangulaire, un agent atomique est visualisé comme un triangle rouge (gris en reproduction monochrome). Un triangle noir représente un agent composé et contient ses sous-agents. La longueur de la base de chaque triangle isocèle est égale à deux fois sa hauteur. Cette condition est nécessaire pour obtenir des triangles qui se décomposent en deux triangles ayant la même forme originale. Une représentation visuelle d'une structure récursive est obligatoirement basée sur l'utilisation d'un objet géométrique ayant cette propriété. Notons que les orientations des triangles représentant les deux sous-agents sont différentes de l'orientation du triangle représentant l'agent composé. Au total, huit orientations différentes sont possibles.

Ajoutons un troisième agent atomique au SMAM de la figure 2. Le SMAM résultant est montré dans la figure 4. En général, dans ce chapitre, nous visualisons les SMAM en utilisant les deux représentations graphiques. Ceci permet aux lecteurs de s'y habituer.

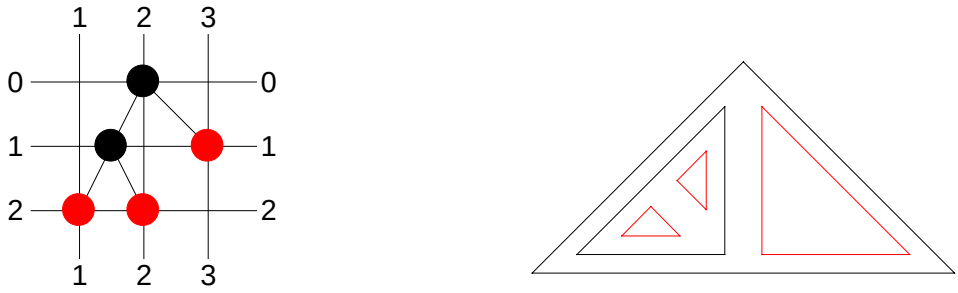


Figure 4 : SMAM composé de trois agents atomiques.

Il n'existe qu'un SMAM unique composé de trois agents atomiques. Il est composé d'un agent composé de deux agents atomiques et d'un troisième agent atomique. Il n'y a pas d'ordre entre les deux agents d'un couple : on ne peut pas identifier un agent « gauche » et un agent « droit » comme on le peut des nœuds d'un arbre binaire. Toutefois, quand nous visualisons, en deux dimensions, les deux sous-agents d'un agent composé, nous sommes obligés de mettre l'un à gauche et l'autre à droite. Afin d'éviter des représentations graphiques différentes d'un même SMAM, nous adaptons la convention que *l'agent ayant la plus grande taille est visualisé à gauche relatif à son partenaire*. Nous présenterons une définition exacte de la mesure de taille dans la section suivante.

L'ajout d'un quatrième agent atomique peut mener à deux SMAM différents. Si l'agent est ajouté au SMAM complet de la figure 4, on obtient le SMAM de la figure 5. Si on combine le nouvel agent atomique avec l'agent atomique numéro 3 de la figure 4, on obtient le SMAM de la figure 6.

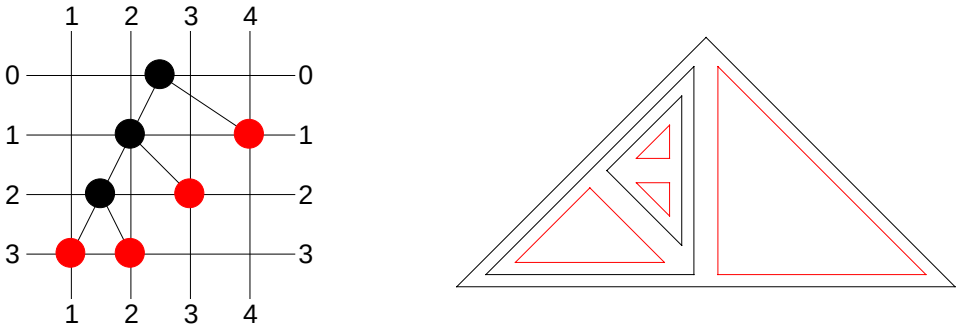


Figure 5 : SMAM composé de quatre agents atomiques (possibilité 1).

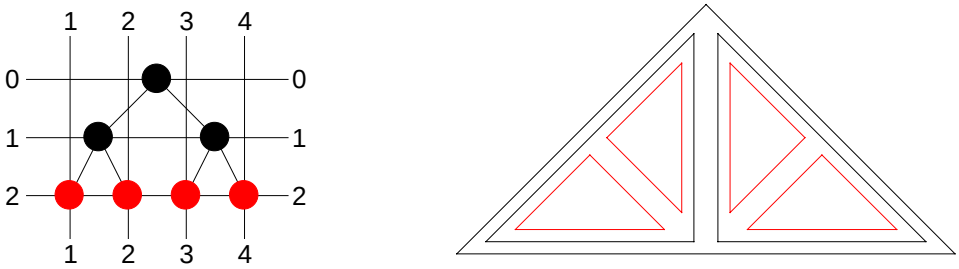


Figure 6 : SMAM composé de quatre agents atomiques (possibilité 2).

Nombre de SMAM différents

Si on ajoute encore plus d'agents, le nombre de SMAM différents réalisables croît d'une manière exponentielle. La table et les graphiques de la page suivante montrent le nombre de SMAM réalisables en fonction du nombre d'agents atomiques. Comme le montre ces données, même à partir d'un nombre limité d'agents atomiques, un grand nombre de SMAM différents peut être construit.

Mathématiquement, le nombre de SMAM différents $N(n)$ comptant n agents atomiques se calcule comme suit :

$$N(n) = \sum_{i=1}^{n/2} N(n-i)N(i)$$

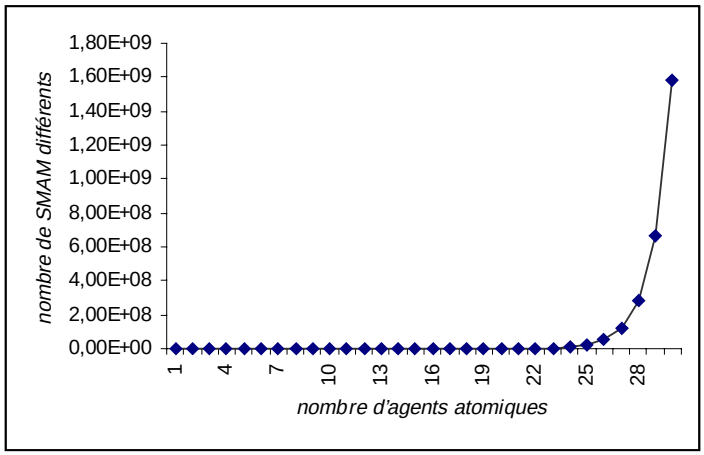
Notons que $N(1) = 1$. La formule précédente est facile à comprendre. Supposons qu'un SMAM C, comptant n agents atomiques, est composé d'un agent A et d'un agent B. Supposons aussi que l'agent A compte autant ou moins d'agents atomiques que l'agent B. Donc, A est composé de minimum 1 et maximum $n / 2$ agents atomiques. B, au contraire, est composé de maximum $n - 1$ et minimum $n / 2$ agents atomiques. Donc, pour un n donné, A peut être structuré de $N(i)$ manières différentes et B de $N(n - i)$ manières différentes, avec i variant entre 1 et $n / 2$. Il suffit de multiplier le nombre de possibilités d'A avec celui de B pour obtenir le nombre de structures possibles de C, et ceci pour un i donné. La sommation de toutes ces possibilités sous i nous donne le nombre total de structures différentes de C.

Nous appelons l'ensemble de tous les SMAM possibles l'ensemble $\mathbb{S}$:

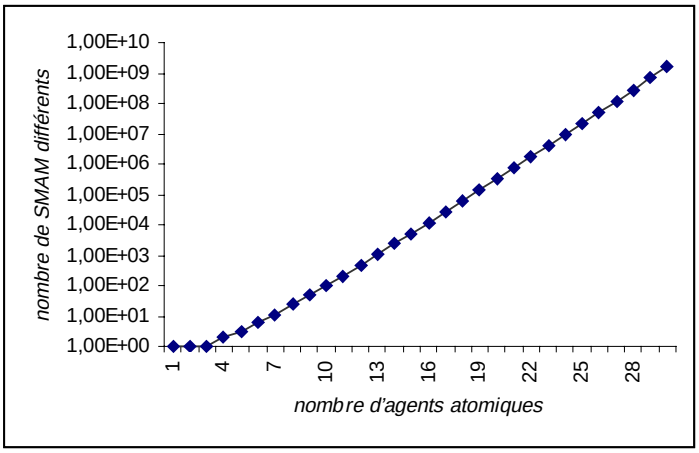
$\mathfrak{S} = \{ S \mid S \text{ est un SMAM} \}$

nombre d'agents atomiques	nombre de SMAM différents
1	1
2	1
3	1
4	2
5	3
6	6
7	11
8	24
9	47
10	103
11	214
12	481
13	1030
14	2337
15	5131
16	11813
17	26329
18	60958
19	137821
20	321690
21	734428
22	1721998
23	3966556
24	9352353
25	21683445
26	51296030
27	119663812
28	284198136
29	666132304
30	1586230523

Tableau 1 : Nombre de SMAM différents.



Graphique 1 : Nombre de SMAM différents.



Graphique 2 : Nombre de SMAM différents (échelle logarithmique).

Structure et organisation

Il est important de réaliser qu'un SMAM n'est pas seulement doté d'une structure, mais qu'il est **égal à sa structure**. Dans le monde abstrait des SMAM, deux entités se distinguent si elles sont structurées différemment. Les agents atomiques, par contre, c'est-à-dire la matière de base, ne peuvent pas être distingués entre eux. Aussi, notons que la structure d'un SMAM définit **l'organisation** des agents, indiquant la manière dont les agents sont groupés :

Un SMAM S est identifié par sa structure, synonyme de son organisation.

Dans plus d'un sens, le concept d'agent atomique reprend la notion ancienne d'atome comme elle a été développée par Leucippus et Democritus : tout est construit avec rien d'autre que des atomes. De ce point de vue, les agents composés constituent un phénomène secondaire lié à la genèse et la destruction de couples. Les SMAM peuvent être considérés comme une pâte à modeler, avec laquelle n'importe quelle structure peut être créée. Toutefois, il existe une distinction importante entre la pâte à modeler de tous les jours et les SMAM : il s'agit d'une pâte *active*. A plusieurs reprises, nous avons souligné la nature active des SMAM. En effet, les agents atomiques et composés sont dotés d'un comportement très spécifique. Ce comportement peut être décrit non seulement en termes d'interactions entre agents, mais également comme un phénomène macroscopique. Toutefois, afin de le formuler au niveau macroscopique, il nous faut des outils pour mesurer les SMAM. Plus loin, nous développerons des mesures très utiles sur les SMAM, mais - dans un premier temps - nous présenterons quelques réflexions sur notre choix de la base deux.

II.2.2 Modéliser en base deux

En général, on peut argumenter que notre approche de base deux est trop restrictive, et qu'il existe des systèmes réels dans lesquels les agents ne sont pas organisés en couples. Toutefois, nous croyons que n'importe quel système peut être vu comme un système organisé d'une manière binaire. De ce point de vue, l'existence de groupes d'agents apparemment non structurés est une illusion créée par la dynamique de leur organisation dans le temps. En effet, si un ensemble d'agents changent continuellement de place dans l'organisation, ils peuvent être aperçus comme non organisés.

La conjecture à la base de notre choix assez restrictif est la suivante : **les interactions entre agents sont toujours des interactions entre deux parties**. S'il existe trois parties, les interactions sont toujours entre une partie et le groupe formé des deux autres. Evidemment, l'organisation des trois peut alterner rapidement, créant l'illusion d'un groupe non organisé. Notons que le concept d'interaction binaire est largement répandu dans les sciences modernes, des modèles de particules physiques aux modèles de communication tel que le modèle de Shannon [Shannon 38].

La conférence

Illustrons l'approche binaire de la structuration des agents en présentant quelques exemples. Le premier exemple est inspiré par la situation où un conférencier s'adresse aux gens dans une salle. La structure du SMAM correspondant est liée à l'organisation « attentionnel » des personnes présentes. Les autres exemples concernent l'organisation opérationnelle d'une « organisation » classique telle qu'une entreprise. Notons que nous n'avons pas encore situé notre concept d'organisation par rapport à la notion d'organisation dans le contexte plus général des Systèmes Multi-Agents. Nous ferons le rapport dans le chapitre II.5 où nous reprendrons les exemples présentés ici qui servent surtout à construire une intuition sur les SMAM.

Etudions d'abord l'exemple d'un conférencier *S* qui répond à une question posée par une personne *Q* dans la salle. Nous formaliserons une organisation possible des personnes présentes du point de vue de leur relation avec le conférencier. Le SMAM correspondant est visualisé dans la figure 7 en utilisant la représentation arborescente. Cette représentation permet une identification facile des agents atomiques. Nous présenterons la représentation triangulaire plus tard.

Imaginons sept personnes dans la salle, le conférencier inclus. A ces sept personnes correspondent sept agents atomiques. A un temps *t*, le conférencier interagit avec son auditoire composé de *Q* et des cinq autres personnes de la salle. De ces cinq personnes, la personne *R* est isolée, peut-être parce qu'elle est fascinée par le discours de *S*. Les quatre autres personnes se trouvent toutes au même niveau de la structure. Toutes ensembles, elles représentent le même intérêt pour *S* que *R*.

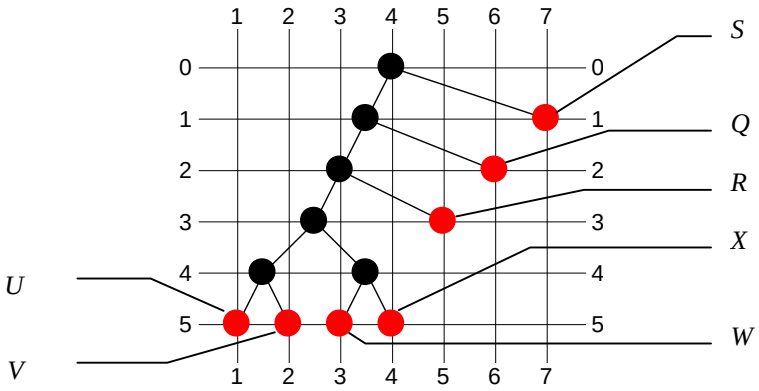


Figure 7 : La conférence (temps *t*).

Notons qu'une telle organisation est, en général, très dynamique. Par exemple, à un temps *t'*, l'organisation des agents peut être très différente :

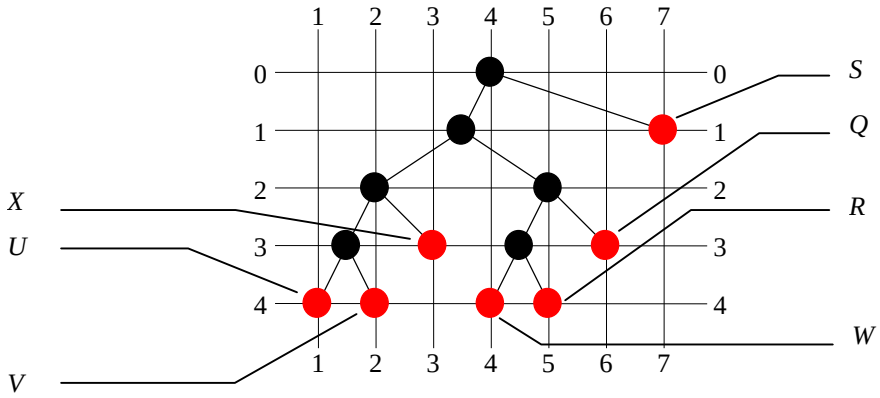


Figure 8 : La conférence (temps *t'*).

Comme le montre la figure 8, *S* occupe toujours une position unique, ce qui est peu surprenant car il s'agit du conférencier. Le public, par contre, s'est divisé en deux groupes de taille identique.

Les figures 9a et 9b montrent les représentations triangulaires des SMAM des figures 7 et 8. La représentation triangulaire montre, mieux que la représentation arborescente, les niveaux relatifs des agents. En effet, le niveau d'un agent correspond directement à l'aire du triangle correspondant. Lorsqu'on donne une interprétation relationnelle aux SMAM, le niveau d'un agents correspond à son importance dans le système.

En général, l'importance relative des agents est plus intéressante que leur importance absolue. Elle nous permet d'estimer l'équilibre du SMAM. Le concept d'équilibre joue un rôle essentiel dans notre modèle. Nous le définirons dans la section suivante. Pour l'instant, remarquons que le SMAM de la figure 9b est plus équilibré

que le SMAM de la figure 9a. Ceci est également reflété dans la différence de profondeur des arbres des figures 7 et 8.

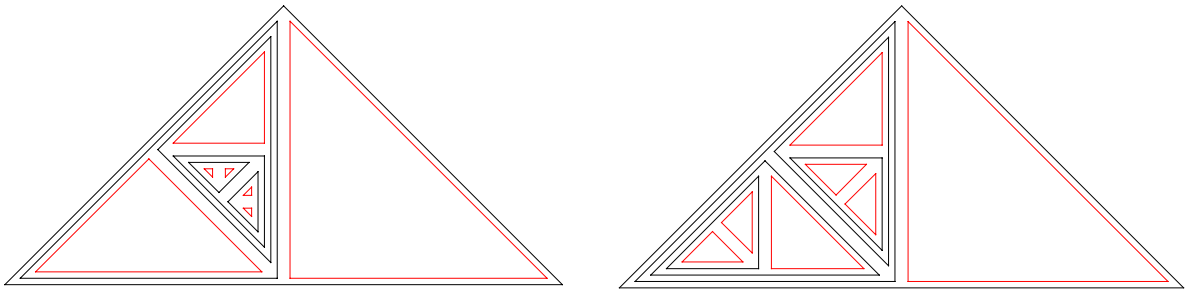


Figure 9a, 9b : La conférence (représentation triangulaire).

Remarquons que les structures des SMAM de l'exemple précédent correspondent à des structures organisationnelles d'un système réel examiné par un observateur. Evidemment, le SMAM même n'est pas un modèle du système réel même. Nous pouvons suivre deux approches fondamentales pour approcher le modèle des SMAM et des systèmes complexes réels. Dans la première approche, la complexité d'un vrai système est modélisée en utilisant un très grand nombre d'agents. Dans la deuxième approche, la nature des agents et de leur comportement est modifiée. En général, nous préférons la première approche, car - dans ce cas - les SMAM utilisés sont des **SMAM purs**, complètement conformes à la théorie. Toutefois, afin d'éviter une trop grande complexité des systèmes résultants, donc pour des raisons pratiques, nous suivons souvent la deuxième approche.

Quelques organisations

Etudions un deuxième exemple qui illustre encore mieux la signification du concept d'organisation dans le contexte des SMAM. Imaginons une entreprise composée de sept personnes : le chef, deux intermédiaires et quatre ouvriers. Ils sont organisés d'une manière hiérarchique comme le montre la figure suivante :

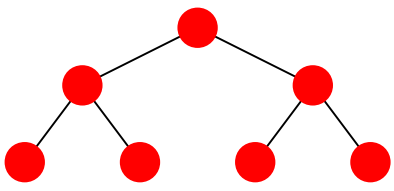


Figure 10 : Hiérarchie.

La figure 10 ressemble superficiellement à la représentation arborescente d'un SMAM. Toutefois, il existe des différences cruciales. D'abord, tous les agents sont représentés par des disques rouges, plutôt que par un mélange de disques rouges et de disques noirs. En effet, seulement les agents atomiques sont visualisés, représentant les membres de l'entreprise. Puis, les arcs ne représentent pas des relations de groupe, mais plutôt des relations d'autorité. Toutefois, nous pouvons facilement concevoir un SMAM duquel la structure correspond à l'organisation de l'entreprise. Il est visualisé dans la figure 11.

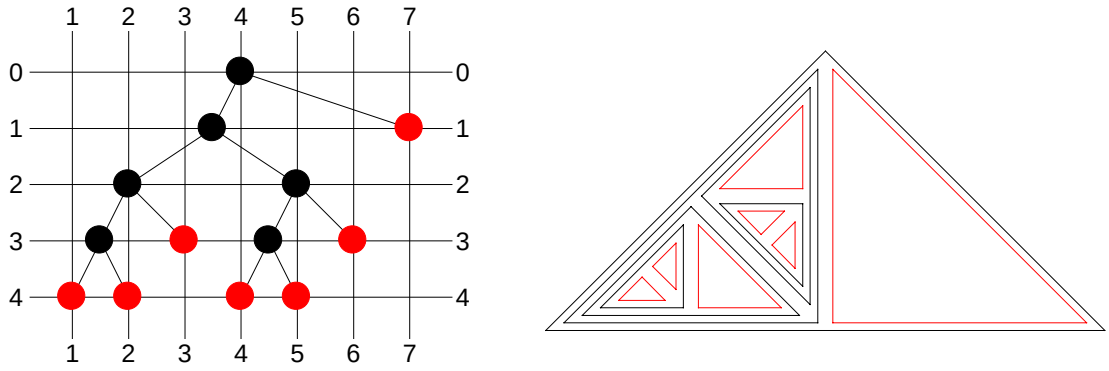


Figure 11 : SMAM correspondant à l'organisation de la figure 10.

Le chef se trouve au niveau 1, les intermédiaires au niveau 3 et les ouvriers au niveau 4. Ils sont tous représentés par des agents atomiques. De plus, nous pouvons identifier les groupes qui constituent l'entreprise et qui sont représentés par des agents composés. L'entreprise complète se trouve au niveau 0 et les deux équipes se trouvent au niveau 2. Aux niveaux 1 et 3 nous trouvons des groupes qui sont rarement identifiés explicitement.

Un groupe non structuré d'agents, comme celui de la figure 12, peut facilement être représenté par un SMAM si le nombre d'agents est une puissance de deux. La figure 13 montre le SMAM correspondant à cette organisation. Il s'agit d'un système parfaitement équilibré.

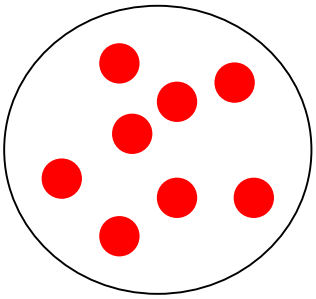


Figure 12 : Groupe non structuré.

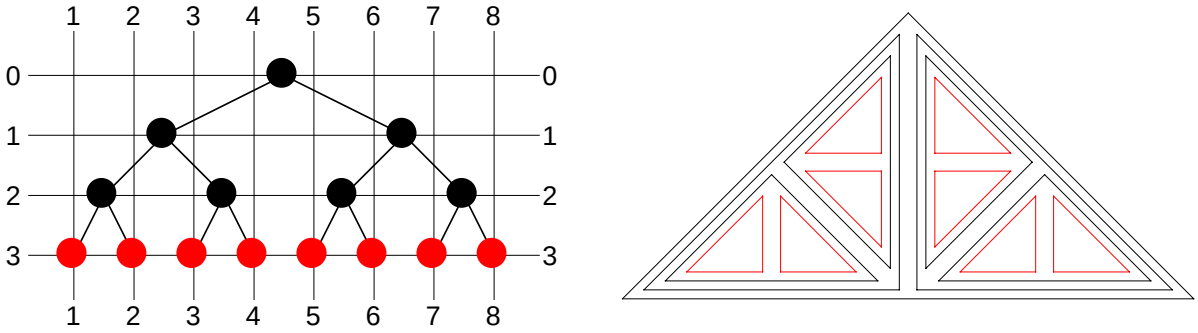


Figure 13 : SMAM correspondant à l'organisation de la figure 12.

Si le nombre d'agents atomiques n'est pas une puissance de deux, il faut prendre en compte la dynamique du système : dans un groupe non structuré, les agents échangent leur position continuellement. Ceci est facile à

visualiser sur un écran informatique (cf. le logiciel FRIENDS Offline, présenté dans le chapitre III.2), mais moins évident à représenter sur papier. Dans la représentation arborescente, nous utilisons une courbe fermée pour indiquer les agents composés qui soutiennent la migration continue de leur sous-agents. Dans la représentation triangulaire, ces agents sont visualisés à l'aide d'un trait plus gras. Ceci est illustré dans la figure 15, qui montre le SMAM correspondant à l'organisation de la figure 14.

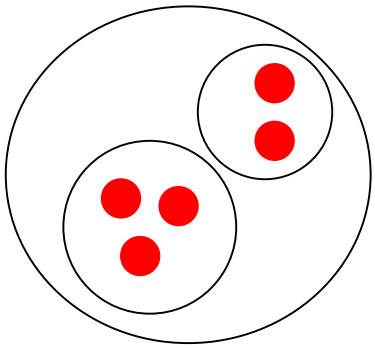


Figure 14 : Organisation non binaire.

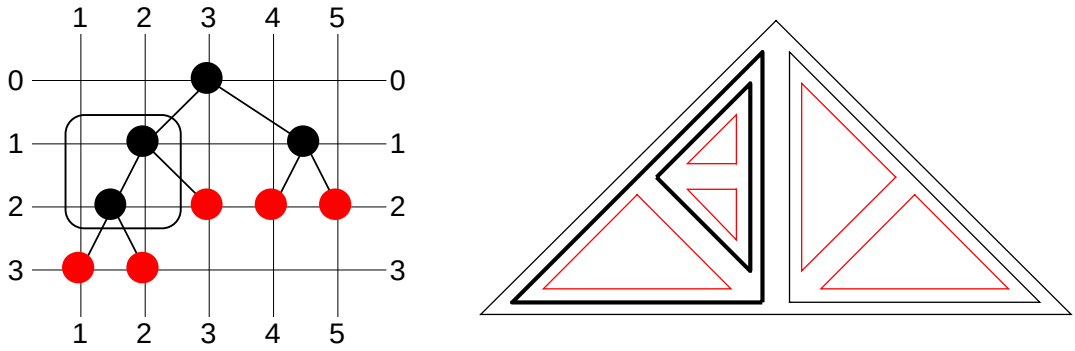


Figure 15 : SMAM correspondant à l'organisation de la figure 14.

Notre dernier exemple concerne une organisation hiérarchique où l'arborescence n'est pas nécessairement d'ordre deux. La figure 16 montre l'organisation que nous aimerions modéliser.

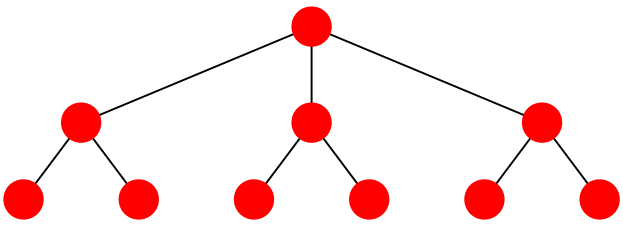


Figure 16 : Organisation hiérarchique non binaire.

La solution se présente naturellement lorsqu'on se rend compte que les trois agents directement dépendant du chef forment un groupe non structuré : ils se trouvent au même niveau hiérarchique. La figure 17 montre la représentation arborescente du SMAM correspondant. La figure 18 montre sa représentation triangulaire.

Comme le montre la figure 18, à partir d'un certain niveau de profondeur, et avec une épaisseur de bordure donnée, il n'y a plus assez d'espace pour laisser des bordures entre les triangles représentant les agents composés

et les triangles représentant ses sous-agents. Cette limitation inévitable rend les agents à ces niveaux plus difficiles à interpréter. En général, la représentation triangulaire donne une bonne vue de l'ensemble, mais elle ne présente pas bien les détails. Pour cette raison, elle est utilisée moins régulièrement que la représentation arborescente.

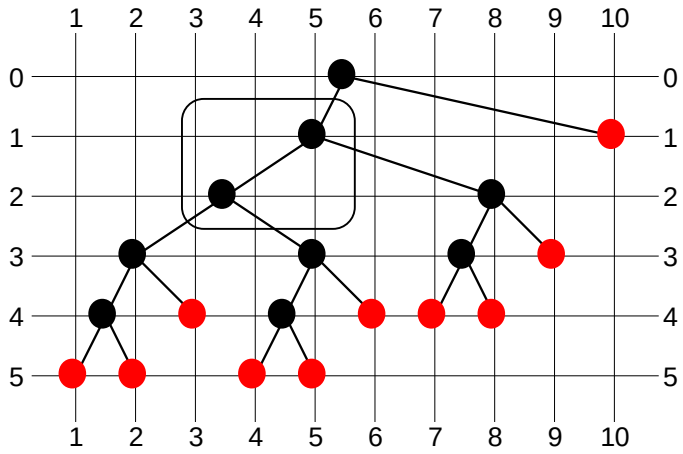


Figure 17 : SMAM correspondant à l'organisation de la figure 16.

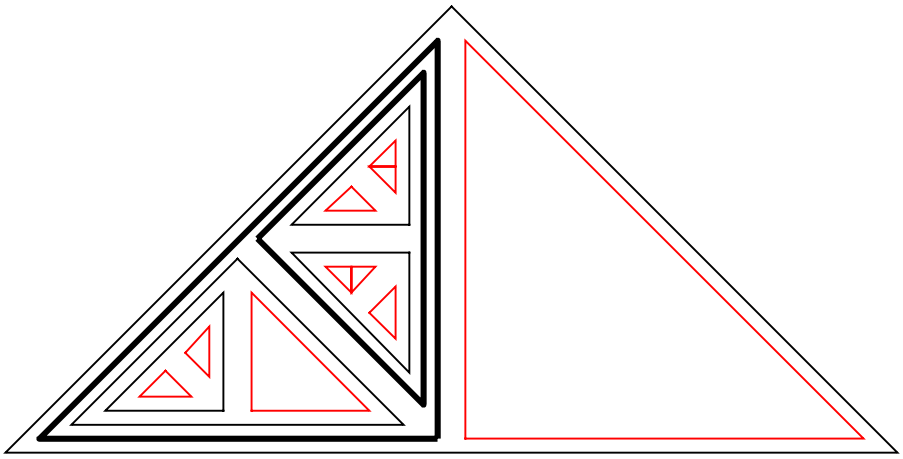


Figure 18 : Représentation triangulaire du SMAM de la figure 17.

En conclusion de cette section, notons que la structure d'un SMAM ne représente que le groupement d'agents. Le modèle n'est pas conçu pour modéliser, par exemple, des relations de pouvoir. Toutefois, la plupart des relations de ce type impliquent implicitement ou explicitement un groupement des agents. Ce groupement d'agents peut, comme nous l'avons illustré, être modélisé assez fidèlement par des SMAM. Nous présenterons une étude plus élaborée de cette correspondance dans le chapitre II.5. Notons, toutefois, que le but des exemples précédents était de développer une intuition pour la nature binaire des SMAM plutôt que de cataloguer des organisations différentes en termes de SMAM.

II.2.3 Nommer les agents

Maintenant que les structures des SMAM n'ont plus de secrets pour nous, nous sommes presque prêts pour étudier les mesures de base des SMAM. Toutefois, nous devons d'abord établir une terminologie fixe des relations entre agents. Supposons, comme illustré dans les figures 19 et 20, que *A* est un agent composé des agents *B* et *C*, et que *B* est composé des agents *D* et *E*.

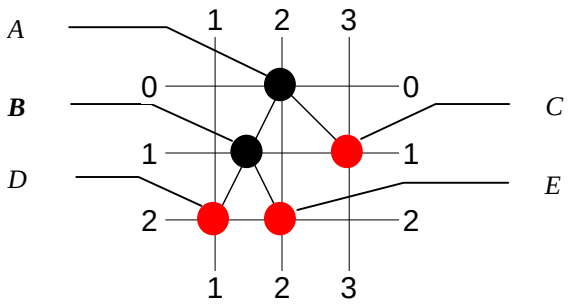


Figure 19 : Nommer les agents.

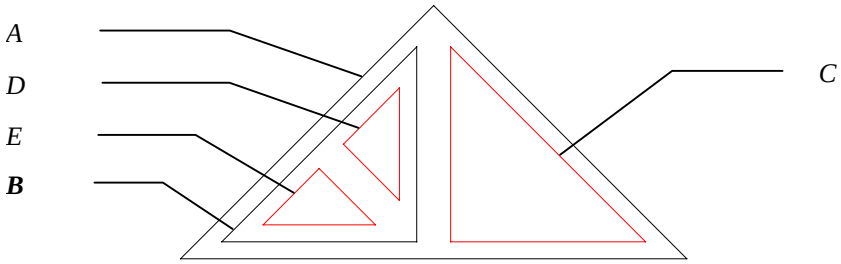


Figure 20 : Nommer les agents (représentation triangulaire).

Pour nommer les relations entre *A*, *B*, *C*, *D* et *E*, nous pouvons utiliser la terminologie des groupes ou celle des arbres. Celle des groupes est plus fidèle au concept des SMAM. Celle des arbres, par contre, est souvent plus facile à utiliser, surtout dans un contexte de programmation. Pour ces raisons, nous utilisons ces deux terminologies dans ce mémoire. Le choix de la terminologie dépend du contexte, et tous les termes du tableau 2 peuvent être utilisés. Toutefois, il faut faire attention de ne pas interpréter incorrectement la terminologie des arbres, car il ne s'agit pas d'arbres binaires, mais de SMAM. Dans le tableau, toute terminologie est relative à l'agent ***B***.

agent	**terminologie des groupes**	**terminologie des arbres**
A	super-agent / couple / groupe	père
B	agent	agent
C	partenaire	frère
D ou E	sous-agent	fils

Tableau 2 : Nommer les agents.

Si un SMAM *A* est un agent d'un SMAM *B*, sans nécessairement être un fils de *B*, nous disons que *A* **fait partie de** *B*. Plus formellement :

> $\forall A, B \in \mathfrak{S}$, *A* fait partie de $B \Leftrightarrow A \subseteq B \Leftrightarrow A = B$ ou *A* fait partie d'un fils de *B*.

Aussi :

> $\forall A, B \in \mathfrak{S}$, *A* fait strictement partie de $B \Leftrightarrow A \subset B \Leftrightarrow$ *A* fait partie d'un fils de *B*.

Les symboles '$\subseteq$' et '$\subset$' sont inspirés par ceux utilisés dans les mathématiques pour indiquer des relations spécifiques entre ensembles. Evidemment, dans le contexte des SMAM, leur sémantique est très différente.

De temps en temps, nous bénéficions d'une description formelle des SMAM. Deux questions se posent : comment présenter un agent atomique et comment présenter un agent composé ?

> Nous présentons formellement **l'agent atomique** comme $\mathbf{\Delta}$ et **l'agent composé** des deux agents *G* et *D* comme **(*G D*)**.

Notons que (*G D*) = (*D G*). Toutefois, lorsque nous utilisons ce formalisme pour décrire un SMAM concret, nous écrivons d'abord le sous-agent ayant la plus grande taille. Par exemple, le SMAM de la figure 21 est décrit comme $(((\Delta\ \Delta)\ \Delta)\ (\Delta\ \Delta))$ et non comme, par exemple, $((\Delta\ \Delta)\ ((\Delta\ \Delta)\ \Delta))$. Si les deux sous-agents ont la même taille, on est libre de les échanger dans la description. Une définition exacte de taille sera donnée dans la section suivante. Nous motiverons notre choix de la représentation « (G D) » dans le chapitre II.4.

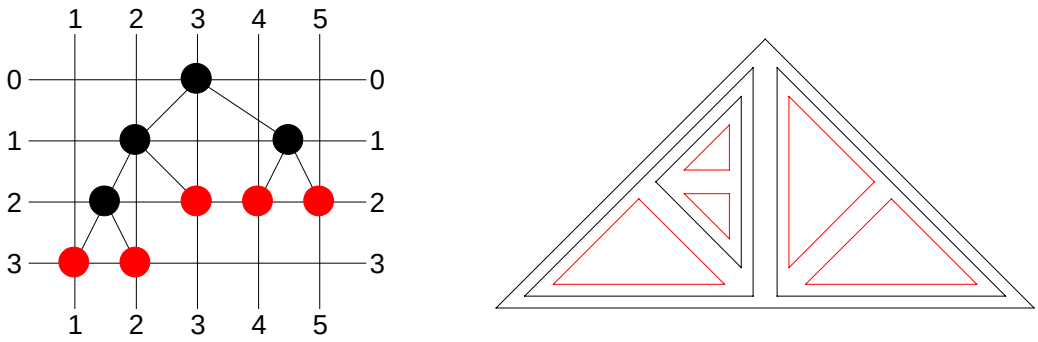


Figure 21 : SMAM simple.

Illustrons le formalisme défini ci-dessus en reformulant la définition de « faire partie de » et de « faire strictement partie de » :

> $\forall A, B \in \mathfrak{S}$, *A* fait partie de $B \Leftrightarrow A \subseteq B \Leftrightarrow A = B$ ou ($B = (G\ D)$ et $(A \subseteq G$ ou $A \subseteq D)$)

Et :

> $\forall A, B \in \mathfrak{S}$, *A* fait strictement partie de $B \Leftrightarrow A \subset B \Leftrightarrow B = (G\ D)$ et $(A \subseteq G$ ou $A \subseteq D)$

II.2.4 Mesurer les SMAM

Etudions un SMAM un peu plus complexe que ceux présentés dans les sections précédentes :

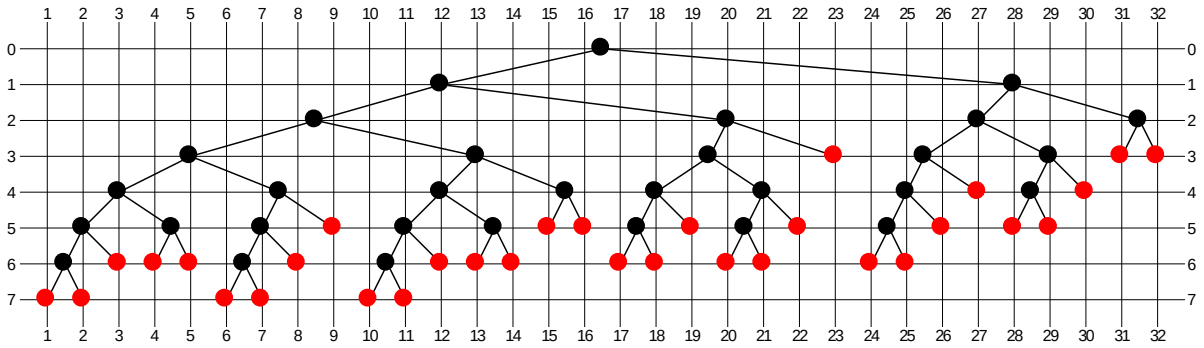


Figure 22 : SMAM complexe.

Notons qu'il s'agit toujours d'un SMAM relativement simple qui n'occupe que quelques bits dans la mémoire d'un ordinateur. Au cœur des applications que nous avons construit se trouvent des SMAM comptant des milliers d'agents atomiques. On peut facilement concevoir des applications basées sur des SMAM de taille 10^6 et plus. Evidemment, dans ces cas, nos deux méthodes de visualisation statique ne sont plus très utiles. Dans la représentation triangulaire, par exemple, les détails deviennent durs à interpréter. La figure 23, qui elle aussi représente le SMAM de la figure 22, donne toujours une bonne vue de l'ensemble, mais ne rend pas bien les niveaux profonds.

Il nous faut des outils d'abstraction qui nous permettent d'extraire l'essentiel d'un SMAM donné. Dans le plus simple (et le plus extrême) des cas, ces outils nous rendront une abstraction sous la forme d'une valeur scalaire, une quantité. Autrement dit, ces outils nous permettent de **mesurer** les SMAM. Dans cette section nous présentons quatre mesures essentielles dont deux indépendantes.

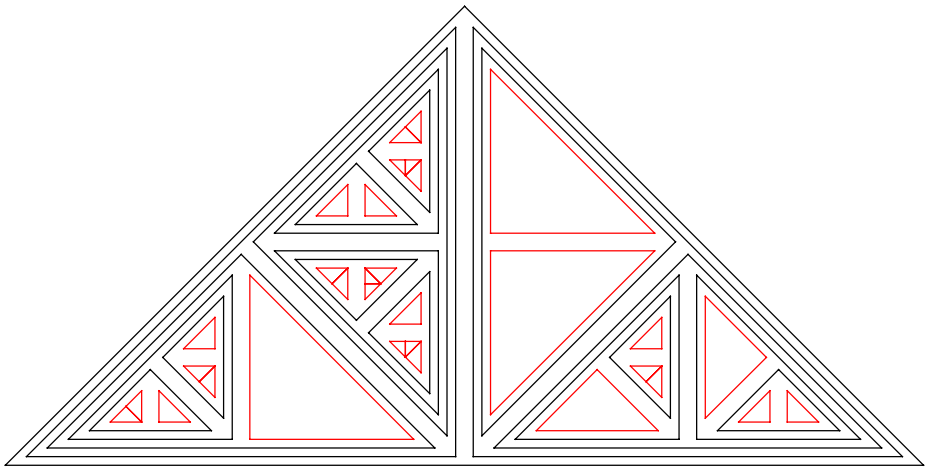


Figure 23 : Représentation triangulaire du SMAM de la figure 22.

Taille et taille réelle

Nous avons déjà rencontré une première mesure : il s'agit de la **taille**. Nous la définissons comme suit :

> La taille $T(S)$ d'un SMAM S est égale à son nombre d'agents.

Parallèlement à cette mesure, nous en définissons une deuxième : la **taille *réelle***. Elle est définie comme suit :

> La taille réelle $TR(S)$ d'un SMAM S est égal à son nombre d'agents atomiques.

Au début de nos recherches sur les SMAM, nous supposions que la relation $T(S)$ / $TR(S)$ pouvait être utilisée pour mesurer certaines propriétés du système. Nous supposions qu'un système comptant moins d'agents composés pour un nombre donné d'agents atomiques serait plus « efficace » qu'un système comptant plus d'agents composés pour le même nombre d'agents atomiques. Il s'avère que la relation entre les deux mesures est complètement indépendante de la structure du SMAM :

> $\forall S \in \mathfrak{S}$, $T(S) = TR(S) + TR(S) - 1 = 2\ TR(S) - 1$

Autrement dit,

> Le nombre d'agents composés d'un SMAM est égal au nombre d'agents atomiques moins un.

Démontrons cette propriété par induction :

- Elle est vraie pour $T(S) = 1$.
- Si elle est vraie pour $T(S) = N >= 1$, elle doit être vraie pour $T(S) = N + 1$, parce que le SMAM doit être composé d'un SMAM de taille $N + 1 - k$ et d'un SMAM de taille k avec $1 <= k <= N$.

  Dans ce cas, le nombre total d'agents composés est égal à $1 + (N + 1 - k - 1) + (k - 1) = N$ parce que $N + 1 - k <= N$ et $k <= N$.

Cette propriété est importante, car elle nous permet d'échanger les termes 'taille' et 'taille réelle' dans un grand nombre de cas. Par exemple, l'expression « la taille d'un SMAM est constante » est équivalante à l'expression « la taille réelle d'un SMAM est constante ».

Notons que cette propriété n'est pas une propriété des arbres binaires. Comme le montre la figure 24, on peut facilement construire un arbre binaire qui compte plusieurs nœuds, mais seulement une feuille.

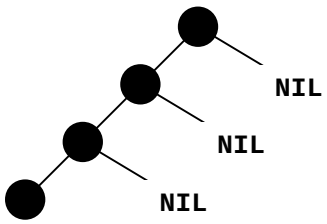


Figure 24 : Arbre binaire comptant une seule feuille.

Equilibre

La taille d'un SMAM constitue une mesure assez évidente. En effet, lorsqu'on essaie de mesurer quelque chose, on commence souvent par la taille. La taille d'un SMAM est liée à la *quantité* (d'agents) du système. Est-ce que nous pouvons trouver une mesure qui est liée à la *qualité* du système ? Puisque les agents atomiques sont tous de la même qualité, une mesure de qualité globale doit être liée à l'organisation des agents. Nous considérons que la qualité d'un SMAM est directement liée à l'équilibre du système. Nous définissons qualitativement l'**équilibre** *EQ*(*S*) d'un SMAM *S* comme suit :

> L'équilibre *EQ*(*S*) d'un SMAM *S* indique jusqu'à quel degré les agents de *S* sont couplés à des agents similaires.

Les figures 25 et 26 montrent deux SMAM de même taille qui sont, respectivement, très bien et très mal équilibrés.

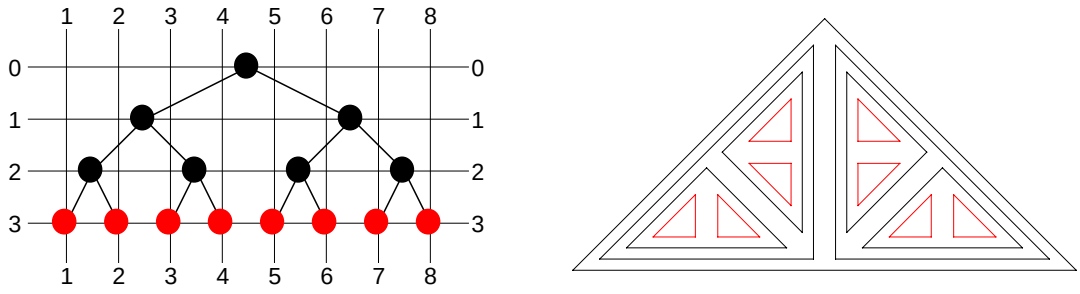


Figure 25 : SMAM très bien équilibré de taille réelle 8.

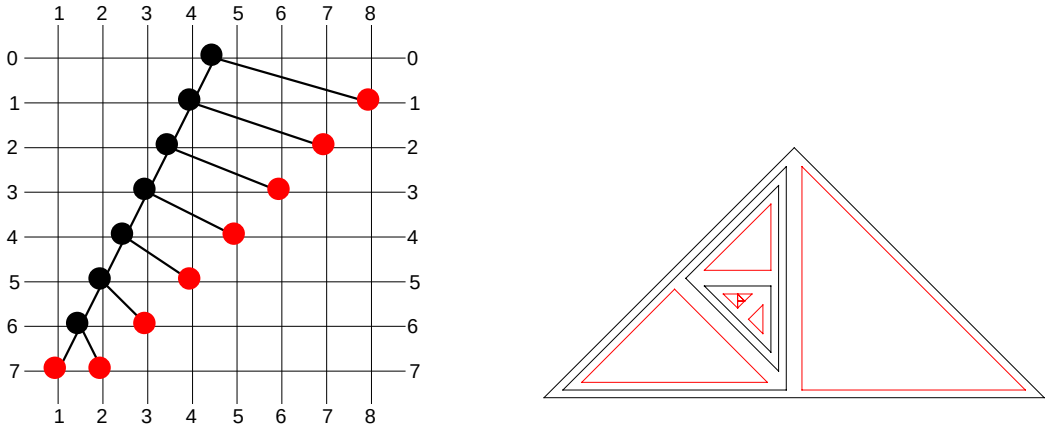


Figure 26 : SMAM très mal équilibré de taille réelle 8.

Dans le SMAM de la figure 25, chaque agent, atomique ou composé, l'agent au niveau 0 exclus, est couplé à un agent identique. Le SMAM est d'un équilibre parfait. Dans le SMAM de la figure 26, tous les agents, les deux au niveau 7 exclus, sont couplés à des agents différents. Surtout aux niveaux près du niveau 0, les différences sont grandes. L'agent atomique au niveau 1, par exemple, est couplé à un agent de taille 13 ! Cette asymétrie est reflétée par les tailles différentes des agents atomiques dans la représentation triangulaire. Si, par contre, l'équilibre est parfait, tous les agents atomiques sont, dans cette représentation, de la même taille.

Entropie

La mesure d'équilibre est, intuitivement ou qualitativement, assez facile à comprendre. Toutefois, nous voulons une mesure quantitative qui nous donne, pour un SMAM donné, un nombre exact. Nous avons trouvé une mesure qui correspond de très près à ce que nous voulons. Elle est dérivée d'une autre mesure qui jouera un rôle important dans notre modèle. Il s'agit de la mesure de l'**entropie**.

La mesure d'entropie est partiellement inspirée par la formule classique de l'entropie comme formulée par Shannon [Shannon 48]. Nous définissons l'entropie $E(S)$ d'un SMAM S comme suit :

$$E(S) = -\sum_{A \in S_\Delta} p_A \log_2(p_A)$$

Dans le contexte des SMAM, p_A représente la probabilité d'atteindre l'agent atomique A en descendant de la racine de l'arbre représentant le SMAM, sachant que la probabilité de choisir le fils 'gauche' est égal à la probabilité de choisir le fils 'droit', soit 0.5. L'expression « $A \in S_\Delta$ » indique que la somme est calculée en prenant uniquement des agents atomiques.

La figure 27 montre un SMAM et les valeurs d'entropie pour tous ses agents. La figure 28 montre les valeurs d'entropie dans les deux systèmes des figures 25 et 26.

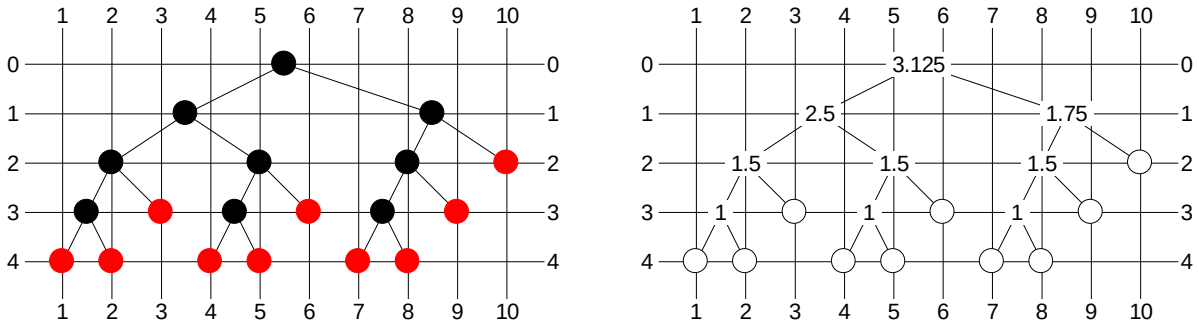


Figure 27 : Valeurs d'entropie des agents d'un SMAM.

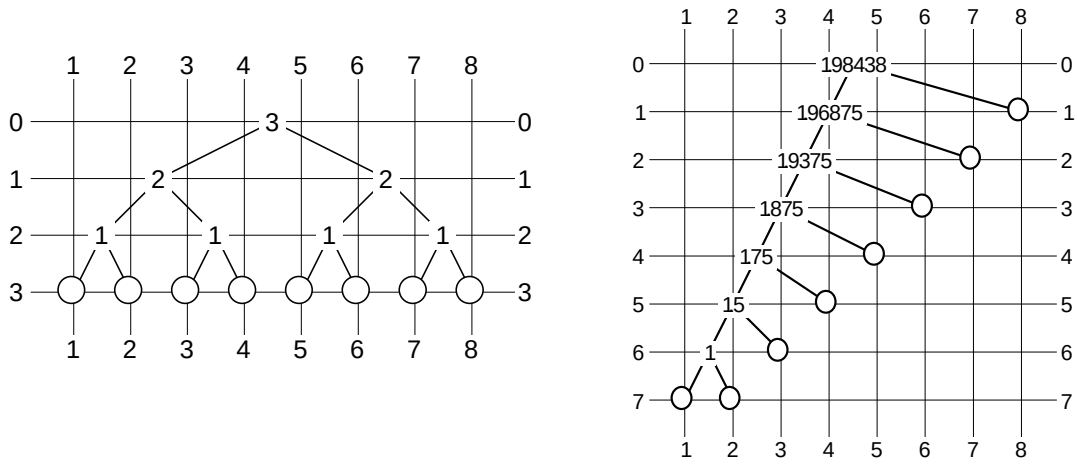


Figure 28 : Valeurs d'entropie dans les SMAM des figures 25 et 26.

Les valeurs de la figure 28 suggèrent que, pour une taille donnée, un SMAM bien équilibré est caractérisé par une entropie plus grande qu'un SMAM mal équilibré. En effet, nous avons établi empiriquement que l'entropie constitue une bonne mesure de la combinaison de taille et d'équilibre. Ceci nous permet de définir quantitativement une mesure d'équilibre comme la suivante :

$$EQ_{exemple}(S) = E(S) / TR(S)$$

Nous définirons cependant une autre mesure. Pour arriver à notre choix final, essayons de spécifier exactement les SMAM comme ceux des figures 27 et 28 représentant les deux équilibres extrêmes. Le SMAM de la figure 27 est tellement parfait que nous l'appelons un **agent atomique d'ordre supérieur**. En général :

Un SMAM S est un agent atomique d'ordre $N \Leftrightarrow TR(S) = 2^N$ et ($(G\ D) \subseteq S \Rightarrow G = D$)

Notons que, selon cette définition, l'agent atomique « de base » est l'agent atomique d'ordre 0. Pour garder une définition facile, nous le considérons, lui aussi, comme faisant partie de la classe des agents d'ordre supérieur, même si son ordre n'est pas supérieur à 0.

Plus loin, nous démontrons que les agents atomiques d'ordre supérieur sont d'une entropie maximale :

Un SMAM S est un agent atomique d'ordre $N \Leftrightarrow E(S) = N = \log_2(TR(S)) \Leftrightarrow E(S) = E_{max}(TR(S))$

$E_{max}(N)$ dénote l'entropie maximale qui peut être atteinte par un SMAM de taille réelle N :

$\forall\ S \in \mathfrak{S},\ TR(S) = N \Rightarrow E(S) \leq E_{max}(N)$

Un autre extrême est illustré par le SMAM de la figure 28, tellement mal équilibré que nous l'appelons un **SMAM linéaire.** En général :

Un SMAM S est un SMAM linéaire $\Leftrightarrow$ ($(G\ D) \subseteq S \Rightarrow (G\ D) = (G\ \Delta)$)

Plus loin, nous démontrons que les SMAM linéaires sont d'une entropie minimale :

Un SMAM S est un SMAM linéaire $\Leftrightarrow$ $S = \Delta$ ou

$$E(S) = \sum_{n=0}^{TR(S)-2} \frac{1}{2^n}$$

$\Leftrightarrow$ $E(S) = E_{min}(TR(S))$

$E_{min}(N)$ dénote l'entropie minimale qui peut être atteinte par un SMAM de taille réelle N :

$\forall\ S \in \mathfrak{S},\ TR(S) = N \Rightarrow E(S) \geq E_{min}(N)$

Les deux extrêmes nous ont motivés à définir quantitativement la mesure d'équilibre comme suit :

$$EQ(S) = \frac{E(S)}{\log_2(TR(S))} \quad \text{si} \quad S \neq \Delta$$

$$EQ(\Delta) = 1$$

Donc, par définition, l'équilibre d'un SMAM parfaitement équilibré, i.e. d'un agent atomique d'ordre supérieur, est toujours égal à 1. A l'autre extrémité du spectre, nous trouvons les SMAM linéaires dont l'équilibre approche 0 pour des grandes tailles. Cette propriété est facile à démontrer :

$$\lim_{TR(S)\to\infty} EQ(S) = \lim_{TR(S)\to\infty} \frac{E(S)}{\log_2(TR(S))} = \frac{\lim_{TR(S)\to\infty} E(S)}{\lim_{TR(S)\to\infty} \log_2(TR(S))}$$

Ou encore, *S* étant un SMAM linéaire, :

$$\lim_{TR(S)\to\infty} EQ(S) = \frac{\lim_{TR(S)\to\infty} \sum_{n=0}^{TR(S)-2} \frac{1}{2^n}}{\lim_{TR(S)\to\infty} \log_2(TR(S))} = \frac{2}{\infty} = 0$$

Donc, la valeur d'EQ(*S*), pour n'importe quel SMAM *S*, se situe toujours dans l'intervalle]0, 1].

Notons que nous n'utilisons la mesure quantitative d'équilibre qu'occasionnellement, car nous sommes, en général, plus intéressés par la mesure combinée de taille et d'équilibre que par celle d'équilibre seul.

Le concept d'agent atomique d'ordre supérieur nous a été utile pour établir une définition quantitative d'équilibre. Puisque nous rencontrons le concept de nouveau dans d'autres sections, il nous semble utile d'introduire maintenant une représentation économique de ces agents dans la représentation arborescente des SMAM. Nous avons choisi, comme symbole représentant un agent d'ordre supérieur, un triangle rouge (ou gris) affichant l'ordre de l'agent. La figure 29 montre, respectivement, les représentations arborescentes compactes des SMAM des figures 25 et 26.

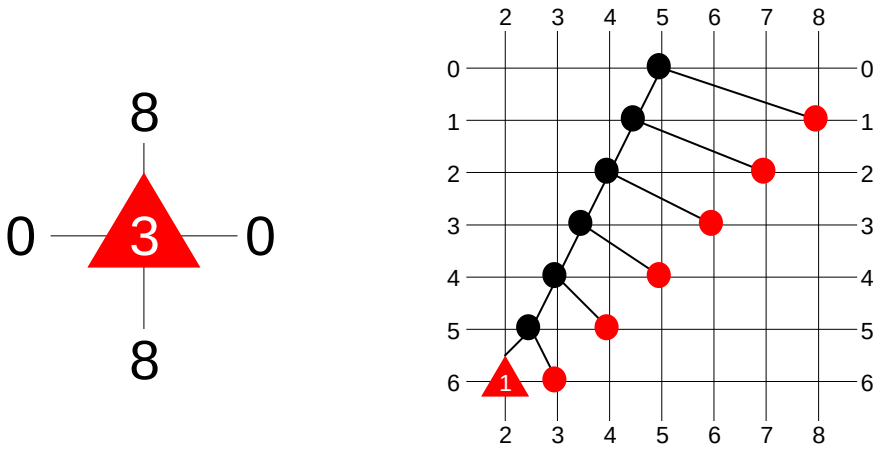


Figure 29 : Représentations compactes des SMAM des figures 25 et 26.

Calculer l'entropie

Nous pouvons exprimer la formule de l'entropie de manières alternatives qui sont souvent plus adaptées au concept des SMAM. A la base de ces expressions alternatives se trouve l'expression de la probabilité de sélection d'un agent atomique *A* en termes de son niveau *N*(*A*) dans la structure. Il est facile de vérifier que la relation est la suivante :

$$p_A = \left(\frac{1}{2}\right)^{N(A)} = \frac{1}{2^{N(A)}}$$

En effet, pour sélectionner un agent *A* au niveau *N*(*A*), il est nécessaire de choisir *N*(*A*) fois soit le fils gauche, soit le fils droit. La probabilité de chaque choix est 1 / 2. La probabilité de la série des *N*(*A*) choix est donc le produit de *N*(*A*) fois 1 / 2.

Notons que, pour tous les agents atomiques *A* d'un SMAM *S*, :

$$\sum_{A \in S_\Delta} \frac{1}{2^{N(A)}} = \sum_{A \in S_\Delta} p_A = 1$$

Maintenant, nous pouvons formuler l'entropie *E*(*S*) d'un SMAM *S* en termes des niveaux des agents atomiques *A* du SMAM :

$$E(S) = -\sum_{A \in S_\Delta} p_A \log_2(p_A) = -\sum_{A \in S_\Delta} \frac{1}{2^{N(A)}} \log_2\left(\frac{1}{2^{N(A)}}\right) = \sum_{A \in S_\Delta} \frac{N(A)}{2^{N(A)}}$$

A partir de cette expression, nous pouvons établir une relation de récurrence entre l'entropie d'un agent composé et les entropies des deux sous-agents. Nous pouvons la déduire mathématiquement assez facilement. Nous utiliserons les symboles *G* et *D* pour dénoter les deux sous-agents du SMAM *S*. *N*'(*A*) dénotera le niveau d'un

agent atomique *A* du SMAM *G*, relatif à *G* et *N'*(*A*) dénotera le niveau d'un agent atomique *A* du SMAM *D*, relatif à *D*. Il est évident que $N(A) = N'(A) + 1$ et que $N(A) = N''(A) + 1$. Donc, nous avons :

$$E(S) = \sum_{A \in S_\Delta} \frac{N(A)}{2^{N(A)}} = \sum_{A \in G_\Delta} \frac{N'(A)+1}{2^{N'(A)+1}} + \sum_{A \in D_\Delta} \frac{N''(A)+1}{2^{N''(A)+1}}$$

L'expression ci-dessus peut être réécrite comme suit :

$$E(S) = \frac{1}{2}\left(\sum_{A \in G_\Delta} \frac{N'(A)}{2^{N'(A)}} + \sum_{A \in G_\Delta} \frac{1}{2^{N'(A)}} \right) + \frac{1}{2}\left(\sum_{A \in D_\Delta} \frac{N''(A)}{2^{N''(A)}} + \sum_{A \in D_\Delta} \frac{1}{2^{N''(A)}} \right)$$

Ou encore :

$$E(S) = \frac{1}{2}\left(\sum_{A \in G_\Delta} \frac{N'(A)}{2^{N'(A)}} + \sum_{A \in D_\Delta} \frac{N''(A)}{2^{N''(A)}} \right) + \frac{1}{2}\left(\sum_{A \in G_\Delta} \frac{1}{2^{N'(A)}} + \sum_{A \in D_\Delta} \frac{1}{2^{N''(A)}} \right) = \frac{E(G)+E(D)}{2} + 1$$

Notons que, derrière la formule apparemment complexe de l'entropie, se cache une relation de récurrence confortablement simple.

Nous avons obtenu deux expressions alternatives de la formule de l'entropie qui s'avèrent très utiles dans la manipulation quotidienne des SMAM. Pour une référence facile, nous les présentons ensemble ci-dessous.

$$E(S) = \sum_{A \in S_\Delta} \frac{N(A)}{2^{N(A)}} \quad \textbf{(1)}$$

$$E(\Delta) = 0 \quad \textbf{(2a)}$$

$$E(\,(G\ D)\,) = \frac{E(G)+E(D)}{2} + 1 \quad \textbf{(2b)}$$

Nous pouvons nous servir de l'expression (1) pour démontrer, d'une manière facile, que l'entropie d'un agent atomique d'ordre supérieur est maximale (pour une taille donnée) et que l'entropie d'un SMAM linéaire est minimale (pour une taille donnée). En effet, (1) nous montre que, puisque $2^{N(A)}$ croît beaucoup plus rapidement que $N(A)$, un SMAM comptant un nombre minimal de niveaux est d'une entropie maximale. Pour une taille réelle égale à une puissance de deux, un tel SMAM est un agent atomique d'ordre supérieur.

De manière similaire, (1) implique qu'un SMAM comptant un nombre maximal de niveaux est d'une entropie minimale. Un tel SMAM est un SMAM linéaire.

La distance entre SMAM comme mesure de différence

Nous pouvons appliquer les mesures que nous avons définies pour comparer des SMAM. Toutefois, nous ne pouvons pas les utiliser pour définir une distance sur l'espace des SMAM, le problème étant que deux SMAM différents peuvent avoir la même taille et équilibre, et donc, la même entropie. Par exemple, comme le montre la figure 31, les deux SMAM de la figure 30, non identiques, ont exactement la même entropie. Donc, une mesure telle que d(*A*, *B*) = | *E*(*A*) - *E*(*B*) | ne satisfait pas la condition que d(*A*, *B*) = 0 implique que *A* = *B*.

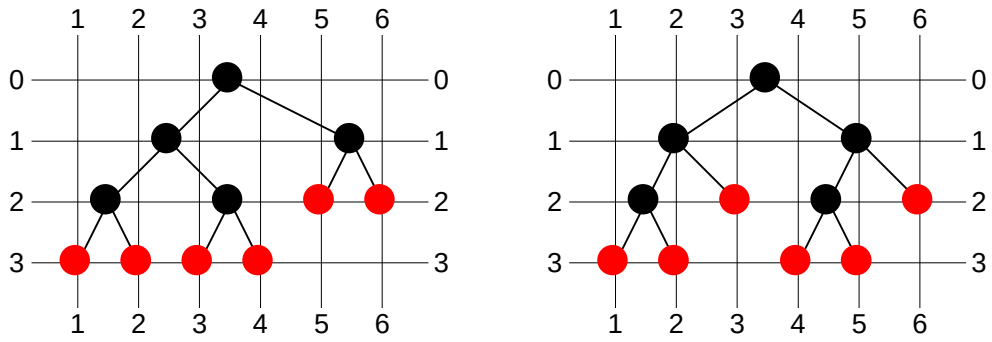


Figure 30 : Deux SMAM différents.

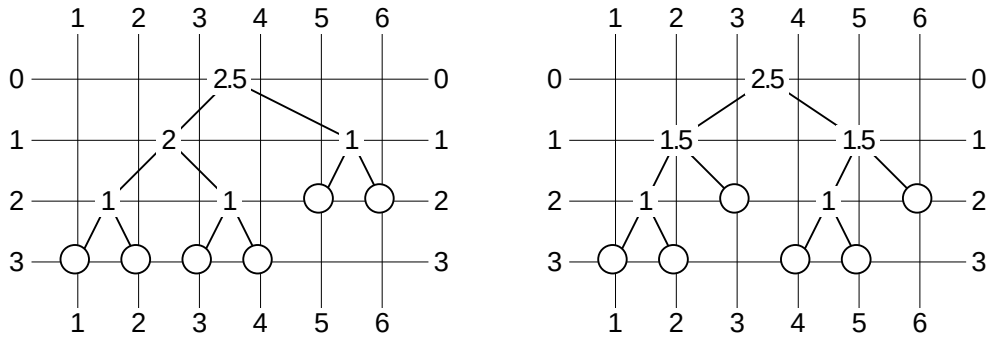


Figure 31 : Valeurs d'entropie des SMAM de la figure 30.

La définition d'une mesure de distance utile pour comparer des SMAM nous semble être un problème non trivial. En général, lorsque nous avons besoin d'une telle mesure, nous utilisons une mesure approximative comme d(*A*, *B*) = | *E*(*A*) - *E*(*B*) |.

La distance entre SMAM comme mesure d'éloignement

Toutefois, nous pouvons définir une mesure de distance très différente en considérant chaque SMAM comme un espace à traverser par des agents migrants. La mesure que nous avons choisie est la suivante :

> La distance d(*A*, *B*) entre deux agents *A* et *B* d'un SMAM *S* est égal à 1a longueur du chemin le plus court entre *A* et *B* dans le graphe de voisinage de *S*.
>
> Le graphe de voisinage d'un SMAM *S* consiste en l'arbre représentant *S* auquel sont ajoutés les arcs entre les deux partenaires de chaque couple.

Les figures 32 et 33 illustrent cette définition. La figure 33 montre le graphe de voisinage du SMAM de la figure 32. Les distances entre l'agent *A* et les autres agents du système sont superposées aux nœuds.

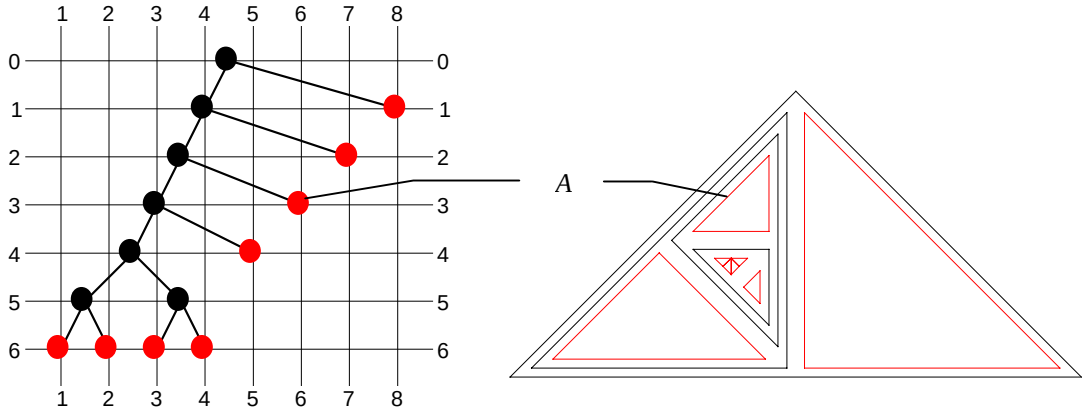


Figure 32 : Agent *A* et ses voisins.

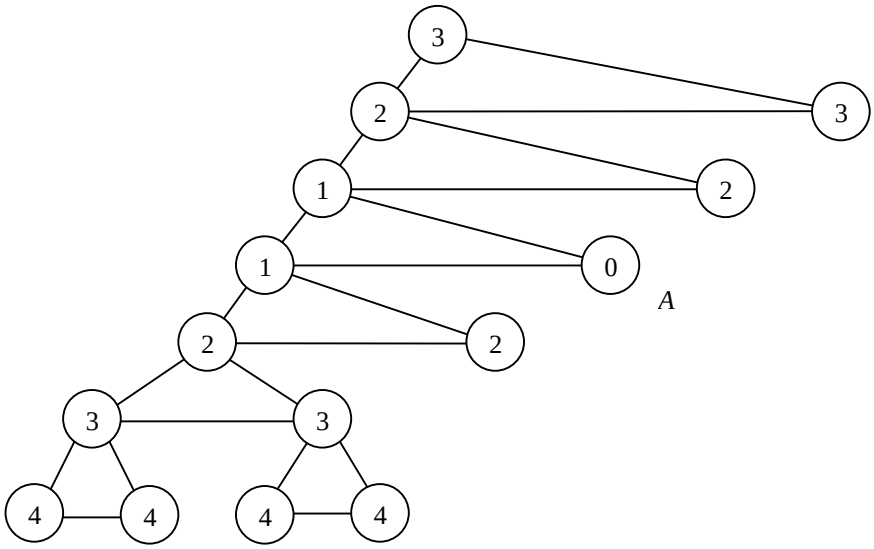


Figure 33 : Graphe de voisinage du SMAM de la figure 32.

Notons que nous avons défini une mesure de distance conforme aux propriétés habituelles. Plus spécifiquement :

Pour tout SMAM *S* et tout *A*, *B*, *C*, agents de *S* :

$d(A, B) = 0 \Leftrightarrow A = B$ (1)

$d(A, B) = d(B, A)$ (2)

$d(A, C) \leq d(A, B) + d(B, C)$ (3)

La validité de ces trois propriétés est une conséquence directe de notre définition de distance.

Comme nous l'avons indiqué, cette mesure peut être utile dans un contexte de migration d'agents. Nous définissons une migration comme suit :

> Une migration est le processus par lequel un agent donné se sépare de son partenaire et se couple à un autre agent.

La distance entre deux agents *A* et *B* d'un SMAM *S* est liée au nombre de migrations élémentaires nécessaires à *A* pour joindre *B*. Nous définissons une migration élémentaire comme suit :

> Une migration élémentaire est *soit* la migration d'un agent vers le frère ou le père de son père, *soit* la migration d'un agent vers un fils de son frère.

Donc, une migration élémentaire permet à un agent de traverser une distance de deux. Toutefois, notons que, en général, la relation entre une distance donnée et le nombre de migrations élémentaires nécessaires pour la traverser n'est pas toujours triviale. Ceci est une conséquence du fait que chaque migration change la structure des agents.

II.2.5 L'évolution naturelle des SMAM

Dans la première section de ce chapitre, nous avons défini les SMAM d'un point de vue structurel. Pour compléter la définition de notre modèle, il nous reste à définir le comportement de ces systèmes. D'abord, nous utiliserons les mesures quantitatives introduites dans la section précédente pour définir l'évolution naturelle des SMAM au niveau macroscopique. Puis, dans la section suivante, nous définirons le comportement des agents au niveau microscopique. Les deux définitions sont qualitativement équivalentes. L'une est le dual de l'autre.

Notons que nous étudions le comportement global des SMAM *isolés*, et non des SMAM en interaction avec un système extérieur non modélisé. Plus spécifiquement, nous définissons l'évolution des SMAM fermés, plutôt que celle des SMAM ouverts. Du point de vue scientifique, nous n'avons pas le choix : un système ouvert peut évoluer dans un sens complètement dirigé par des facteur externes et donc non modélisés. Evidemment, notre choix ne nous empêche pas d'étudier des SMAM ouverts dans le contexte d'un SMAM fermé enveloppant.

Définissons d'abord, en termes exactes, la notion d'évolution. Dans le temps, par les actions des agents, un SMAM *A* peut se transformer, en une ou plusieurs étapes, en un SMAM *B*. Puisque nous travaillons dans le monde abstrait des SMAM, nous ne pouvons pas utiliser la notion de temps physique dans nos définitions. Comme notion de temps, nous utilisons le nombre d'actions effectuées par les agents. Nous définissons une **action** comme suit :

> Une action d'un SMAM *S* est , *soit* la prise de décision de migrer plus la migration, *soit* la prise de décision de ne pas migrer.

Nous définissons une **action interne** comme suit :

> Une action interne d'un SMAM *S* est une action effectuée par un des SMAM faisant partie de *S*.

Maintenant, nous somme prêts pour introduire la notion de **transformation** d'un SMAM **en un nombre donné d'actions internes** :

$\forall A, B \in \mathfrak{S}$, N un nombre naturel, $TRANS_N(A) = B \Leftrightarrow A$ se transforme, en exactement N actions internes, en B.

Aussi :

$\forall A, B \in \mathfrak{S}$, N un nombre naturel, $p(TRANS_N(A) = B)$ est la probabilité que A se transforme, en exactement N actions internes, en B.

Nous définissons la notion d'**évolution naturelle** d'un SMAM comme suit :

$\forall A, B \in \mathfrak{S}$, N un nombre naturel, A évolue naturellement vers $B \Leftrightarrow A \rightarrow B \Leftrightarrow$
$\lim_{N \to \infty} p(TRANS_N(A) = B) = 1$

Maintenant que nous avons défini la *notion* d'évolution, nous pouvons définir l'évolution elle-même. Notons que, lorsque nous étudions l'évolution d'un SMAM, nous considérons le système total comme **fermé** (et non **ouvert**) dans le sens suivant :

Un SMAM fermé est un SMAM vers lequel aucun agent ne peut immigrer et à partir duquel aucun agent ne peut émigrer.

Un SMAM ouvert est un SMAM vers lequel des agents peuvent librement immigrer et à partir duquel des agents peuvent librement émigrer.

Nous appelons l'ensemble de tous les SMAM fermés l'ensemble $\mathfrak{SF}$:

$\mathfrak{SF} = \{ S \mid S$ est un SMAM fermé $\}$

Nous appelons l'ensemble de tous les SMAM ouverts l'ensemble $\mathfrak{SO}$:

$\mathfrak{SO} = \{ S \mid S$ est un SMAM ouvert $\}$

Notons que les agents faisant strictement partie d'un SMAM S fermé peuvent être ouverts. Ceci nous permet d'étudier les SMAM ouverts dans un contexte contrôlé.

Evolution de la taille

Comme nous voulons construire un modèle minimal, l'évolution des SMAM doit être simple à définir. Aussi, elle doit être la plus universelle possible, au moins d'une manière intuitive. Ces critères ne sont pas faciles à satisfaire et tout choix est nécessairement personnel. Notre choix se décompose en deux éléments. Le premier est lié à la taille des SMAM et définie une constante plutôt qu'une évolution :

La taille d'un SMAM fermé est constante dans le temps.

Plus exactement :

$\forall A, B \in \mathfrak{SF}$, N un nombre naturel, $TRANS_N(A) = B \Rightarrow T(A) = T(B)$

Notons que, à cause de la correspondance entre taille et taille réelle, nous pouvons formuler la propriété ci-dessus comme suit :

La taille réelle d'un SMAM fermé est constante dans le temps.

Ou, plus exactement :

$\forall A, B \in \mathfrak{SF}$, N un nombre naturel, $TRANS_N(A) = B \Rightarrow TR(A) = TR(B)$

Autrement dit, le nombre d'agents atomiques d'un SMAM fermé est conservé dans le temps. Les agents atomiques ne peuvent pas émerger du néant ou disparaître sans laisser des traces. Notons qu'il y a une correspondance intéressante entre notre définition et la première loi de la thermodynamique. Selon la première loi, l'énergie totale d'un système isolé est conservée, ou - en exploitant la relation entre masse et énergie - la masse totale d'un système isolé est conservée. Nous étudierons cette correspondance en plus de détail dans le chapitre II.5.

Notons que, pendant une période importante de nos recherches, nous avions défini d'une manière différente l'évolution naturelle de la taille des SMAM. Spécifiquement, nous avions défini que les SMAM ont la tendance naturelle à *maximiser* leur taille [Van Aeken 98a] [Van Aeken 98b] [Van Aeken 98c]. Dans ce contexte, nous parlions - implicitement - d'une classe spécifique de systèmes ouverts. Nous étudierons en détail ces systèmes dans le chapitre II.5 lorsque nous situerons notre modèle par rapport à la Vie Artificielle. Nous montrerons que l'évolution de ces systèmes est une conséquence de l'évolution plus générale définie dans ce chapitre-ci.

Evolution de l'équilibre

Le premier composant de l'évolution naturelle des SMAM est lié à la taille des systèmes. Le deuxième est lié à l'équilibre des SMAM. Intuitivement, un couple composé de deux agents très différents n'est pas très stable et se décompose facilement. La recherche d'un certain équilibre nous semble être une caractéristique importante d'une grande classe de systèmes. Ceci nous a mené à déclarer que :

L'équilibre d'un SMAM fermé se maximise dans le temps.

Plus exactement :

$\forall A, B, S \in \mathfrak{SF}$, $T(A) = T(S), A \rightarrow B \Rightarrow EQ(S) \leq EQ(B)$

Pour des SMAM S différents de Δ, nous avons quantitativement défini $EQ(S)$ comme $E(S) / \log_2(TR(S))$. Autrement dit, pour des S différents de Δ, $E(S) = EQ(S) \log_2(TR(S))$. Donc, une formulation équivalente de notre définition de l'évolution de l'équilibre des SMAM fermés est la suivante :

L'entropie d'un SMAM fermé se maximise dans le temps.

Ou, plus exactement :

$\forall A, B, S \in \mathfrak{SF}, T(A) = T(S), A \rightarrow B \Rightarrow E(S) \leq E(B)$

Notons la correspondance entre cette propriété et la deuxième loi de la thermodynamique déclarant également que l'entropie d'un système fermé se maximise. Nous explorerons cette correspondance plus en détail dans le chapitre II.5.

Illustrons l'évolution naturelle des SMAM à l'aide d'un exemple. Prenons le SMAM *S* à un temps *t* affiché (avec entropies) dans la figure 34. A un temps $t + \Delta t$, il sera transformé en le SMAM de la figure 35. Notons que notre définition ne spécifie pas *dans les détails* comment les SMAM évoluent. Nous étudierons cette question dans le prochain chapitre dédié aux algorithmes à la base du comportement des agents. Une évolution possible de l'entropie de *S pourrait* être celle affichée dans le graphique 3.

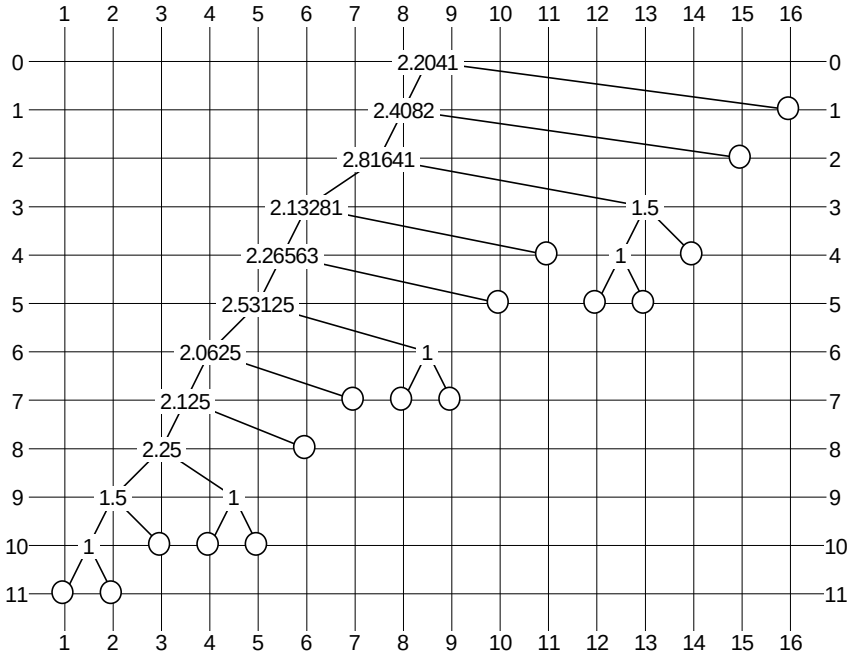


Figure 34 : SMAM *S* au temps *t*.

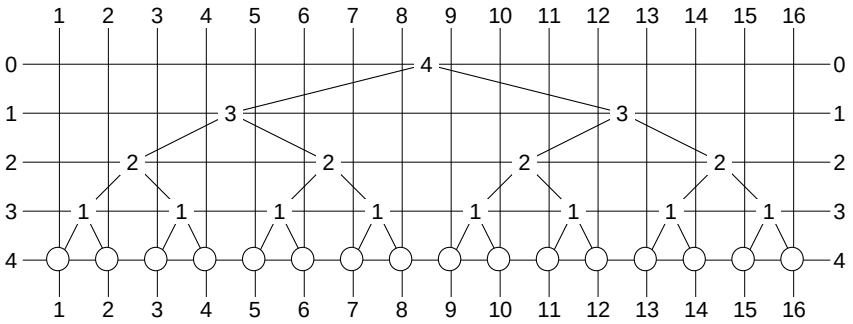
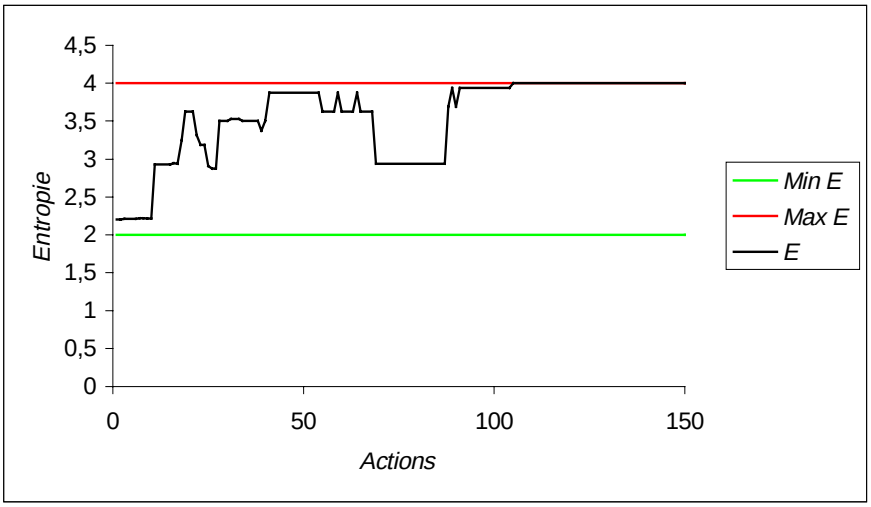


Figure 35 : SMAM *S* au temps $t + \Delta t$.



Graphique 3 : Evolution possible de l'entropie du SMAM de la figure 34.

II.2.6 Le comportement des agents

Dans la section précédente, nous avons défini l'évolution des SMAM fermés à un niveau macroscopique. Reste à savoir comment ce comportement macroscopique se traduit par des interactions microscopiques entre les agents.

Soulignons d'abord que les interactions de base entre agents dans un SMAM sont des migrations. Par définition, le monde des SMAM ne contient rien que des SMAM. Il n'existe pas d'objets extérieurs sur lesquels les agents peuvent agir.

L'augmentation naturelle de l'équilibre implique, qualitativement et par définition, que les agents seront de plus en plus couplés avec des agents similaires. Ceci nous a motivé à définir le comportement de base suivant :

> Les agents d'un SMAM interagissent en suivant le principe « qui se ressemble, s'assemble ».

En suivant ce mécanisme, un agent couplé à un agent non identique, essayera de trouver un meilleur partenaire en migrant vers un endroit différent du SMAM. Par définition, ce mécanisme supporte l'évolution d'un système fermé vers un état d'équilibre maximal. Puisque la migration ne crée et ne détruit pas d'agents atomiques, la taille réelle d'un système fermé est constante. En conséquence, la taille elle aussi est constante : lorsqu'un agent migre, un couple existant est brisé et un couple nouveau est formé.

Notons que le comportement défini n'exige des agents non seulement des capacités de migration, mais également des capacités de perception et d'apparence. Nous étudierons en détail ces éléments dans le chapitre II.5.

Comme dans la définition macroscopique, il s'agit d'une définition générale qui ne détaille pas les actions. Le détails des migrations est défini par l'algorithme ou les algorithmes à la base du comportement des agents. Nous en spécifierons un certain nombre dans le chapitre suivant. D'ailleurs, si on souhaite démontrer quantitativement la correspondance entre la définition macroscopique et la définition microscopique, on est obligé de spécifier l'algorithme utilisé, ce qui implique une démonstration au sens microscopique → macroscopique. En général, dans un contexte pratique, ce qui nous intéresse est de savoir si un algorithme donné, implémentant le

mécanisme « qui se ressemble, s'assemble », mènera un SMAM *S* de taille fixe vers un état d'équilibre, et donc d'entropie, maximale. Nous trouverons des démonstrations de ce type dans le chapitre suivant qui est dédié aux algorithmes à la base du comportement des agents.

II.2.7 Les définitions essentielles des SMAM

Concluons ce chapitre en compilant les définitions essentielles des SMAM :

$\mathfrak{S} = \{ S \mid S = \Delta$ ou $S = (G\ D)$ avec $G, D \in \mathfrak{S} \}$

$\forall A, B \in \mathfrak{SF}$, N un nombre naturel, $TRANS_N(A) = B \Rightarrow T(A) = T(B)$

$\forall A, B, S \in \mathfrak{SF}$, $T(A) = T(S)$, $A \rightarrow B \Rightarrow E(S) \leq E(B)$

CHAPITRE II.3

ALGORITHMES CENTRALISES ET DISTRIBUES

Inequality is the cause of all local movements.
Leonardo da Vinci

II.3.1 L'algorithme et sa classification

David Marr, célèbre chercheur en Intelligence Artificielle, distinguait trois niveaux dans les systèmes informatiques (*information-processing systems*) : le niveau conceptuel (*computational theory*), le niveau algorithmique (*representation and algorithm*) et le niveau d'implémentation (*hardware implementation*) [Marr 82]. Cette classification est acceptée par un grand nombre de chercheurs : elle semble tellement naturelle. En effet, un concept, le tri par exemple, peut être réalisé à l'aide d'un certain nombre d'algorithmes différents. A un niveau plus bas, un algorithme peut être implémenté de plusieurs manières différentes. Il n'est même pas nécessaire que la plate-forme physique soit un ordinateur numérique : certains algorithmes sont implémentés à l'aide de constructions en bois [ibid.].

Les idées de Marr étaient basées sur la conception classique de l'informatique dans laquelle un système informatique calcule des sorties à partir des entrées. La notion de la machine de Turing illustre bien cette conception. Toutefois, dans un monde de plus en plus interactif et orienté multimédia, cette conception classique semble, dans certains cas, dépassée. Prenons, par exemple, la boucle d'événements dans un système de gestion moderne. Dans ce cas, nous n'avons plus le simple flux entrées → calcul → sortie. De plus, dans beaucoup de systèmes modernes, la fonctionnalité d'un concept est dépendante de l'algorithme utilisé ou de l'implémentation effectuée. Souvent, le temps de réponse et l'utilisation des ressources, deux facteurs liés à l'algorithme et son implémentation, sont directement liés à la fonctionnalité. Prenons l'exemple d'un jeu vidéo : s'il s'exécute deux fois plus lentement, ce n'est plus le même jeu. Dans un certain sens, beaucoup de nos systèmes actuels sont des systèmes temps réel. Dans ce contexte, les trois niveaux de Marr sont moins orthogonaux que dans le contexte classique.

Concept, algorithmes et implémentations des SMAM

En ce qui concerne les Systèmes Multi-Agents Minimaux, est-ce que nous pouvons isoler le concept, les algorithmes et les implémentations ? Notons d'abord que les SMAM sont des systèmes abstraits, dépourvus de toute fonctionnalité dans le sens classique. Toutefois, pour des raisons de recherche, nous pouvons essayer de construire des systèmes qui ressemblent le plus possible à la notion théorique, c'est-à-dire des **simulations** des SMAM. Aussi, nous aimerions bien appliquer le concept abstrait et construire des **applications** des SMAM. Donc, la question à poser est plutôt la suivante : est-ce que nous pouvons séparer le niveau conceptuel, algorithmique et implémentationnel d'une simulation ou d'une application des SMAM ?

La définition des SMAM spécifie leur évolution à long terme, plutôt que leur évolution à court terme. Notre choix d'algorithme est libre, pourvu qu'il implémente l'évolution à long terme des SMAM réalisés. Dans ce sens, le niveau conceptuel et le niveau algorithmique sont assez orthogonaux. Notons toutefois que dans les simulations, c'est exactement l'évolution à court terme qui nous intéresse : l'évolution à long terme, nous la connaissons déjà. En ce qui concerne les applications des SMAM, l'évolution à court terme est encore plus importante : en général, les utilisateurs veulent des résultats dans les délais les plus courts possibles. Donc, le niveau algorithmique est bien séparé du niveau conceptuel des SMAM théoriques, mais toutefois lié au niveau conceptuel des SMAM simulés et appliqués. Ceci implique que nous présentons, dans ce chapitre, des informations qui concernent non seulement le niveau algorithmique, mais également la conception des SMAM simulés et appliqués.

Comme le niveau algorithmique, le niveau d'implémentation est séparé du niveau conceptuel des SMAM théoriques. Par exemple, dans le monde des SMAM, le temps est mesuré en actions, et non en secondes. Si ces actions sont exécutées d'une manière sérielle, parallèle, rapidement ou lentement, le modèle ne le spécifie pas. En ce qui concerne les simulations et les applications, la situation est différente. Par exemple, un algorithme parallèle doit être sérialisé pour qu'il puisse tourner sur une machine dotée seulement d'un processeur. Dans un contexte applicatif, une machine lente nécessite souvent des algorithmes plus efficaces qu'une machine puissante. Dans les deux cas, nous sommes obligés de spécifier, jusqu'à un certain degré, la plate-forme physique que nous utilisons.

Les algorithmes présentés dans ce chapitre sont conçus pour être exécutés sur une architecture sérielle ou sur une architecture parallèle. Toutefois, pour des raisons pratiques, les algorithmes distribués sont pourvus d'un mécanisme de sérialisation. Lorsque l'on veut implémenter ces algorithmes sur une architecture parallèle, on doit vérifier que le mécanisme de sérialisation est une bonne simulation du parallélisme réel.

Pour faire abstraction de la plate-forme, comme c'est l'habitude dans l'étude d'algorithmes, nous avons indiqué les temps d'exécution en actions (comme définis dans le chapitre II.2) et non en secondes.

Dans la section suivante, nous présentons et évaluons quatre algorithmes représentatifs. D'autres ont été développés, mais ils sont, en général, moins intéressants et moins performants. Notons, toutefois, qu'il s'agit d'algorithmes simples, servant surtout à l'illustration des principes. Dans la partie « applications » du mémoire, nous présenterons deux autres algorithmes, basés sur le concept de SMAM *augmenté*, défini ultérieurement.

Critères de classification

Avant de présenter les algorithmes, essayons d'abord d'en spécifier une classification. Cette classification peut être effectuée d'un point de vue algorithmique ou d'un point de vue « agent ». Dans ce chapitre, nous nous concentrons sur l'aspect algorithmique, sans négliger la dimension « agent », plus élaborée dans le chapitre II.5. Nous pouvons identifier les critères de classification suivants :

L'algorithme peut être **centralisé ou distribué**. Un algorithme centralisé est conçu pour un seul processeur ayant accès, à tout instant, à la structure de données complète. Un algorithme distribué, par contre, est conçu pour une architecture parallèle (physique ou virtuelle) et souvent ne nécessite que l'accès aux données locales. Notons que seuls les algorithmes distribués correspondent directement aux comportements individuels des agents d'un SMAM.

Au niveau conceptuel, l'algorithme peut être simple ou complexe. Les algorithmes distribués simples correspondent à des agents réactifs. Les algorithmes distribués complexes, par contre, correspondent à des agents cognitifs effectuant, dans un sens ou un autre, une forme de planification. Nous considérons la **simplicité conceptuelle** des algorithmes comme une qualité désirable.

Au niveau de la complexité algorithmique aussi, les algorithmes peuvent être simples ou complexes. Une action d'un algorithme simple nécessite un temps de calcul qui croît, par exemple, d'une manière linéaire avec la taille du SMAM. Le rapport peut être non linéaire ou même exponentiel dans le cas d'un algorithme complexe. Pour des raisons évidentes, nous préférons la **simplicité algorithmique**.

Que l'algorithme garantisse une **maximisation de l'entropie** ou pas, constitue un critère extrêmement important. Si cela n'est pas le cas, le système informatique utilisant l'algorithme ne simule pas le modèle des SMAM. Notons que surtout les algorithmes locaux risquent de se bloquer dans des optima locaux.

Certains algorithmes maximisent l'entropie seulement si le SMAM initial peut être transformé en un agent atomique d'ordre supérieur. D'autres, par contre, fonctionnent mieux si le SMAM initial est moins régulier. La **robustesse de l'algorithme** par rapport à la nature du SMAM initial constitue un critère important.

Atteindre l'optimum est une chose, le faire rapidement en est une autre. La **vitesse de convergence** nous intéresse beaucoup.

Certains algorithmes maximisent l'entropie, mais oscillent légèrement autour de l'optimum. D'autres commencent à osciller avant avoir atteint l'optimum, ce qui est beaucoup plus grave. L'**absence d'oscillations** est un autre critère que nous considérons.

Par définition, les algorithmes centralisés utilisent des ressources globales. Par contre, en ce qui concerne les algorithmes distribués, dans un contexte « agent », nous pouvons nous poser les questions suivantes :

- Est-ce que l'algorithme nécessite un champ de perception non local ?
- Est-ce que l'algorithme nécessite un champ de migration non local ?
- Est-ce que l'algorithme nécessite un champ de communication non local ?

Notons que le concept de champ est développé - entre autres - dans [Van Aeken 90a] [Van Aeken 90b] [Baeijs 96] et [Ferrand 94].

La **localité des ressources nécessaires** constituent un dernier critère. Nous préférons les algorithmes qui ne nécessitent que des ressources locales.

II.3.2 Présentation et évaluation des algorithmes

Dans cette section, nous présentons et évaluons quatre algorithmes différents. Nous les identifions par leur noms utilisés dans FRIENDS Offline : « global aggregated », « local aggregated », « global E-matcher II » et « local E-matcher ».

Nous évaluons chaque algorithme en utilisant trois SMAM aléatoires de taille réelle 32 et trois SMAM aléatoires de taille réelle 31. Les premiers ont le potentiel de se transformer en agents atomiques d'ordre 5. Les derniers sont des SMAM qui, même dans leur état d'entropie maximale, sont toujours dotés d'une structure relativement irrégulière.

Dans notre évaluation, les SMAM initiaux sont choisis aléatoirement à l'intérieur de l'ensemble des SMAM mal équilibrés, c'est-à-dire des SMAM ayant un équilibre inférieur à 0.25.

La figure 1 montre un exemple d'un SMAM initial de taille 32. La figure 2 représente sa transformation optimale. La figure 3 représente un SMAM initial possible de taille 31. La figure 4 représente sa transformation souhaitée. Notons que, dans cette section dédiée à une analyse plus formelle, nous visualisons les SMAM en affichant les entropies et en utilisant la représentation compacte des agents atomiques d'ordre supérieure.

Pour chaque algorithme, et pour chaque cas (SMAM de taille réelle 32 ou 31), nous présentons, en utilisant la mesure d'entropie, trois traces typiques de l'évolution du SMAM initial vers le SMAM final.

Notons que nous identifierons les algorithmes par leurs noms anglais, utilisés dans FRIENDS Offline, notre plate-forme d'expérimentation (voir le chapitre III.2).

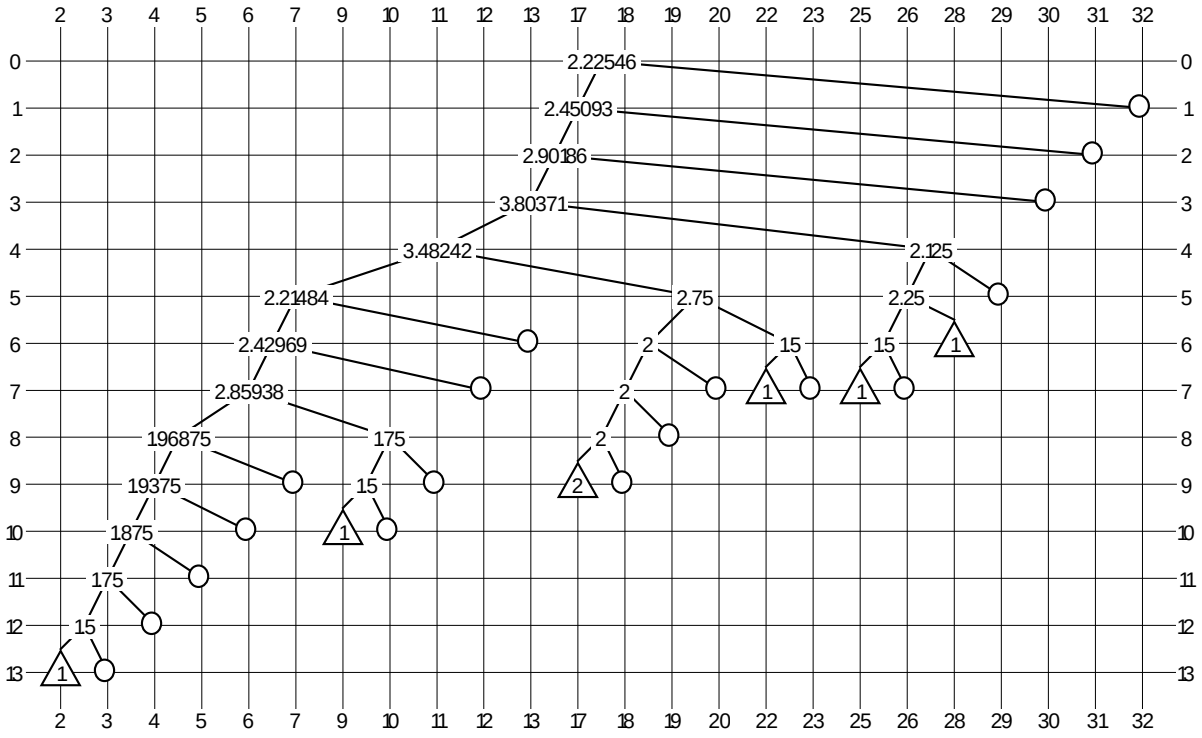


Figure 1 : SMAM de taille réelle 32.

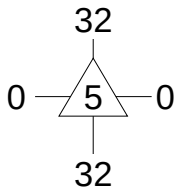


Figure 2 : Transformation optimale du SMAM de la figure 1.

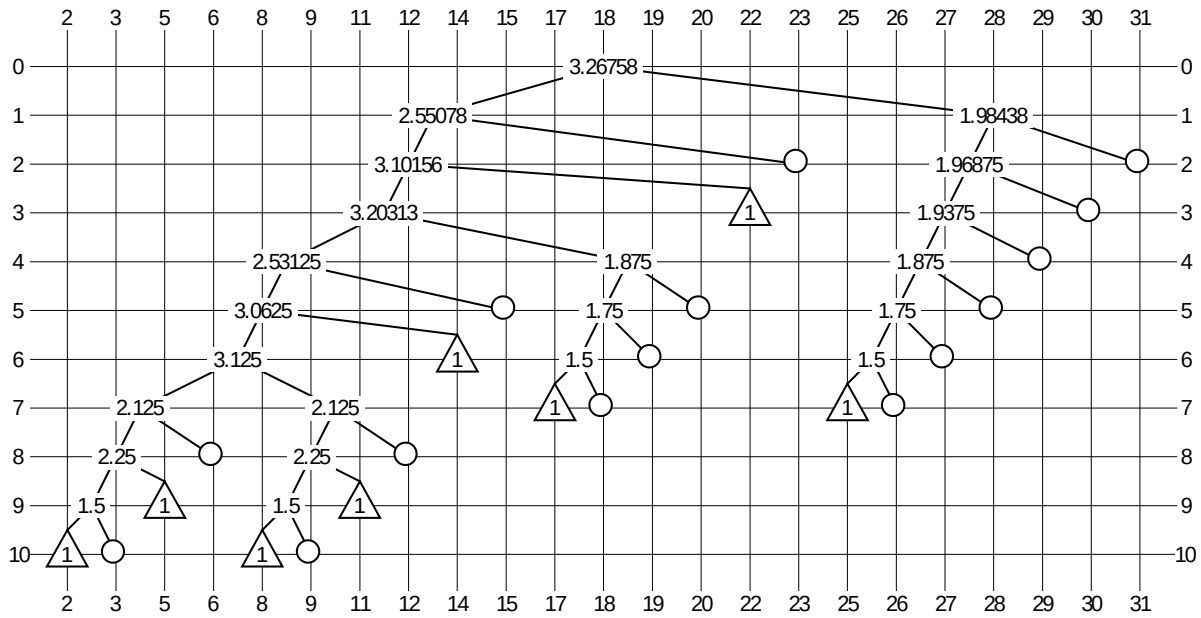


Figure 3 : SMAM de taille réelle 31.

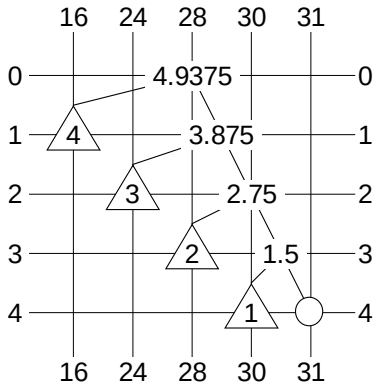


Figure 4 : Transformation optimale du SMAM de la figure 3.

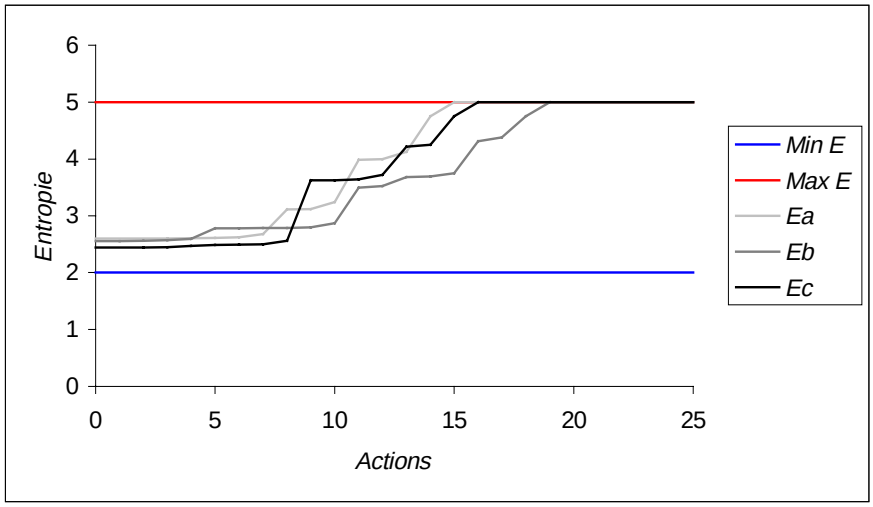
L'algorithme « global aggregated »

Les deux premiers algorithmes sont basés sur une agrégation non réversible des agents. Les agents forment des agents atomiques d'ordres de plus en plus supérieurs. Jamais, en utilisant cet algorithme, un agent atomique d'ordre supérieur n'est décomposé.

L'algorithme est défini (pour *une* action) comme suit :

1) `Créer une liste de tous les agents atomiques d'ordre supérieur non couplés à des agents identiques.`
2) `Trouver, dans la liste, deux agents du même ordre. La liste est cherchée en ordre (d'agent atomique) ascendante.`
3) `S'il n'y a pas deux candidats à coupler, abandonner.`
4) `Faire migrer le premier candidat vers le deuxième.`

Le graphique 1 montre l'évolution, guidée par l'algorithme « global aggregated », de trois SMAM différents de taille 32.



Graphique 1 : Comportement de l'algorithme « global aggregated » pour *TR*(*S*) = 32.

Notons que l'algorithme mène, pour des SMAM d'une taille réelle égale à une puissance de deux, à une convergence très rapide. En effet, l'algorithme n'est pas seulement très simple, mais - dans des cas bien précis - aussi très efficace.

De cet algorithme, qui implémente le comportement microscopique des SMAM, i.e. « qui se ressemble, s'assemble », nous pouvons facilement démontrer qu'il soutient l'évolution macroscopique des SMAM, i.e. qu'il garanti la maximisation de l'équilibre. Plus spécifiquement, nous pouvons démontrer qu'un SMAM *S* de taille réelle $TR(S) = 2^N$ (*N* entier), évolue, en un nombre fini d'actions, vers un état d'équilibre maximal, *si* les agents migrent en s'agrégeant en des agents atomiques d'ordre supérieur.

Notons que cette démonstration sert surtout comme exemple, illustrant comment la correspondance microscopique ⟶ macroscopique du modèle des SMAM peut être démontrée dans des cas spécifiques. Nous ne la répéterons pas pour les autres algorithmes.

La démonstration par induction est :

— Le nombre d'agents atomiques d'ordre 0 est 2^N.

— Si le nombre d'agents atomiques d'ordre i est 2^{N-i}, i différent de N, l'algorithme mène, en $2^{N-(i+1)}$ actions ou moins, à l'existence de $2^{N-(i+1)}$ agents atomiques d'ordre $i + 1$.

Si 2^{N-i} est différent de 1, il existe un nombre pair d'agent atomiques d'ordre i.

Donc, si un agent atomique d'ordre i se trouve couplé à un agent différent, il existe un autre agent atomique d'ordre i dans le même cas.

Donc, les agents d'ordre i peuvent migrer et se coupler jusqu'à ce qu'ils soient tous couplés en $2^{(N-i)/2} = 2^{N-(i+1)}$ agents atomiques d'ordre $i + 1$.

— Si le nombre d'agents atomiques d'ordre i est 2^{N-i}, i égal à N, le système est parfaitement équilibré,

parce que le système compte 2^N agents atomiques et contient un agent d'ordre N.

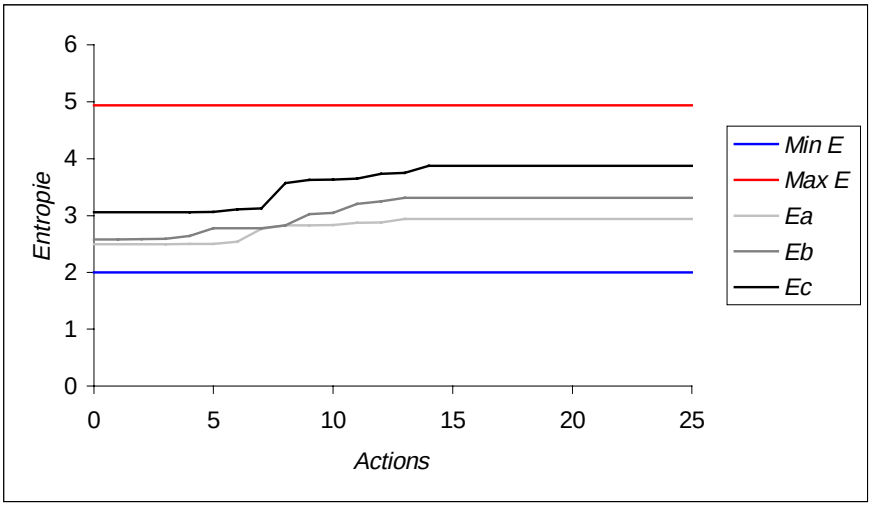
Le nombre total d'actions nécessaires pour évoluer d'un état initial vers un état d'équilibre parfait, dans le pire des cas, est donné par l'équation suivante :

$$\#actions_{max} = \sum_{i=0}^{N-1} 2^{N-(i+1)} = \sum_{j=0}^{N-1} 2^j < 2^N$$

Donc, le nombre maximal d'actions à effectuer est toujours inférieur au nombre d'agents atomiques dans le système.

Notons que l'équilibre du système non seulement converge vers sa valeur maximale, mais semble monter d'une manière monotone. Ce comportement est lié aux détails de notre implémentation et n'est pas une caractéristique absolue de l'algorithme.

Etudions maintenant le comportement de cet algorithme pour des SMAM d'une taille différente d'une puissance de deux. Le graphique 2 montre l'évolution d'entropie pour trois SMAM différents de taille réelle 31.



Graphique 2 : Comportement de l'algorithme « global aggregated » pour *TR*(*S*) = 31.

La convergence de l'algorithme est toujours rapide, mais n'atteint, en aucun cas, le résultat attendu. Donc, dans ce cas spécifique, il ne simule pas l'évolution naturelle des SMAM. Comment expliquer ce comportement lamentable ? Regardons deux SMAM finaux (par rapport à l'évolution guidée par l'algorithme) pour mieux comprendre ce phénomène :

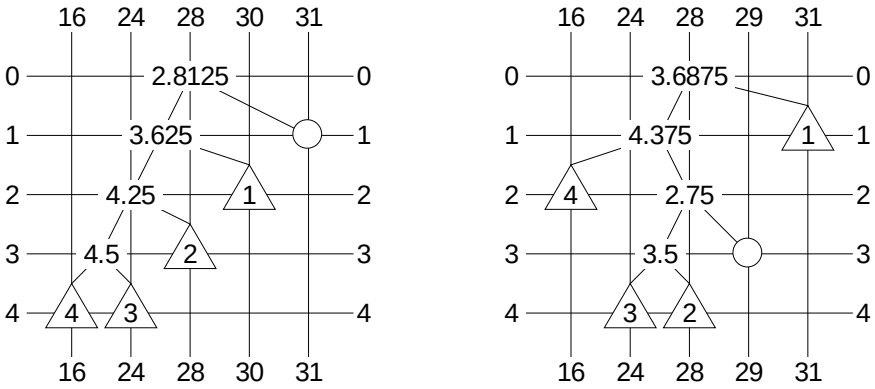


Figure 5a, 5b : SMAM mal équilibrés.

La figure 5 illustre bien le fait que les SMAM arrêtent d'évoluer une fois que plus aucun agent atomique d'ordre supérieur ne peut être formé. La manière dont les agents atomiques d'ordre supérieur sont groupés est un aspect complètement négligé par l'algorithme : un agent atomique d'ordre important peut être couplé à un agent atomique d'ordre peu important. Donc, il est possible que le système évolue vers les SMAM de la figure 5, très mal équilibrés et très différents du SMAM optimal de la figure 4.

L'essence du problème est que l'algorithme ne manipule que des agents « parfaits », des agents atomiques d'ordre supérieur. Un algorithme vraiment fiable doit être capable de faire migrer d'autres SMAM aussi. Nous présenterons de tels algorithmes plus tard. Toutefois, dans un premier temps, étudions une version distribuée de l'algorithme « global aggregated ».

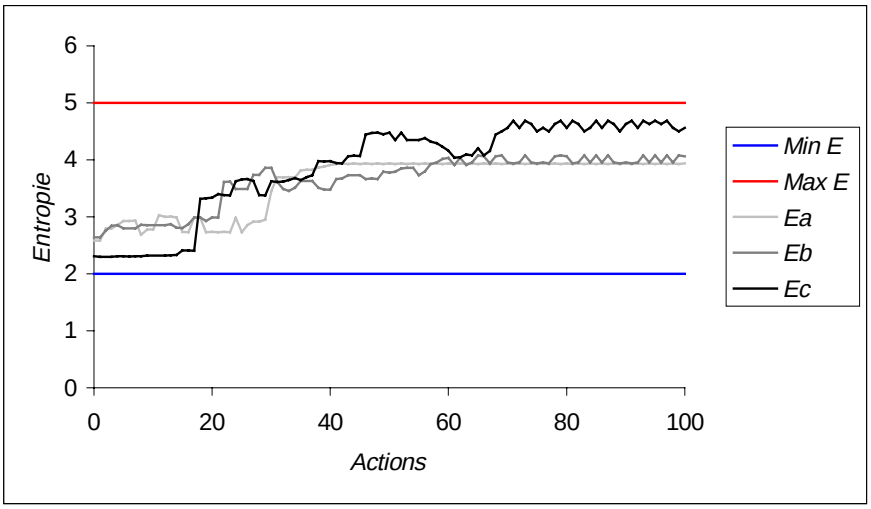
L'algorithme « local aggregated »

L'algorithme « local aggregated » est également basé sur l'agrégation systématique des agents en des agents atomiques d'ordres de plus en plus supérieurs. Il est différent de l'algorithme précédent, les actions sont effectuées d'une manière distribuée : toute interaction est locale.

L'algorithme est défini (pour *une* action) comme suit :

1) Créer une liste de tous les agents atomiques d'ordre supérieur non couplés à des agents identiques.
2) Eliminer, dans la liste, tous les agents dont le partenaire est un agent atomique d'ordre supérieur.
3) Abandonner si la liste filtrée est vide.
4) Sélectionner, aléatoirement, dans la liste, un candidat. Ceci simule le parallélisme.
5) Chercher, dans le partenaire du candidat, un sous-agent identique au candidat. S'il n'existe pas, ou s'il y en a deux, le sous-agent 'gauche' est choisi.
6) Faire migrer le premier candidat vers le deuxième.

Le graphique 3 montre l'évolution, guidée par l'algorithme « local aggregated », de trois SMAM différents de taille 32.



Graphique 3 : Comportement de l'algorithme « local aggregated » pour *TR*(*S*) = 32.

La convergence de ce deuxième algorithme est plus lente que celle du premier. Par conséquence, nous avons dû adapter l'échelle de l'axe des abscisses. En général, ce qui est peu surprenant, les algorithmes distribués sont moins efficaces que les algorithmes centralisés.

Ce qui est plus grave est le fait que, dans les cas illustrés, l'équilibre parfait n'est jamais atteint. L'algorithme ne simule pas l'évolution théorique des SMAM. Ce comportement décevant est lié, entre autres, au fait que - en suivant l'algorithme - les agents n'explorent que des agents aux niveaux plus bas. Nous appelons un tel algorithme *descendant*.

Pour nous, l'intérêt principal de l'algorithme est qu'il illustre bien le phénomène des oscillations. Les oscillations surviennent lorsque deux agents ou plus migrent continuellement entre eux. Si seulement deux agents sont impliqués, le résultat est une **oscillation simple** (voir la figure 6 et le graphique 4). Si plus d'agents sont impliqués, l'**oscillation** est **complexe** (voir la figure 7 et le graphique 5).

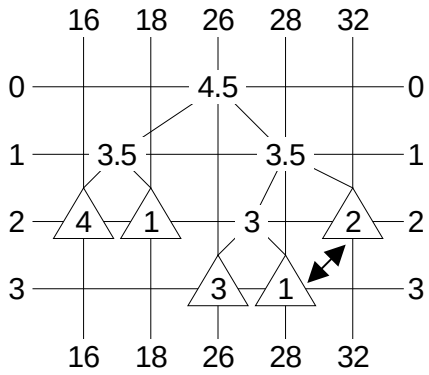
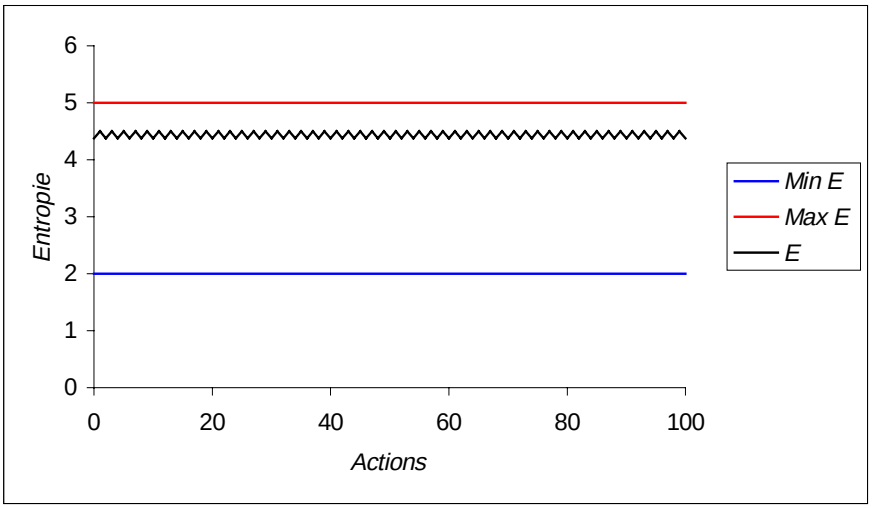


Figure 6 : Oscillation simple.



Graphique 4 : Oscillation simple.

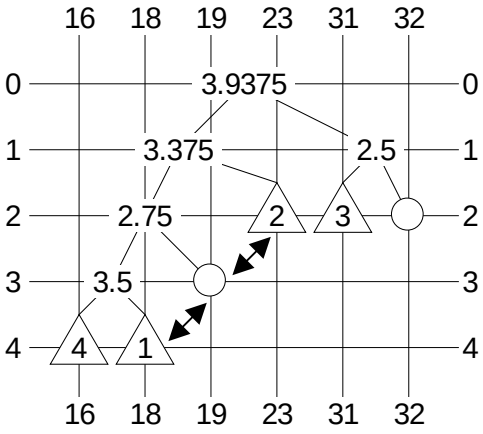
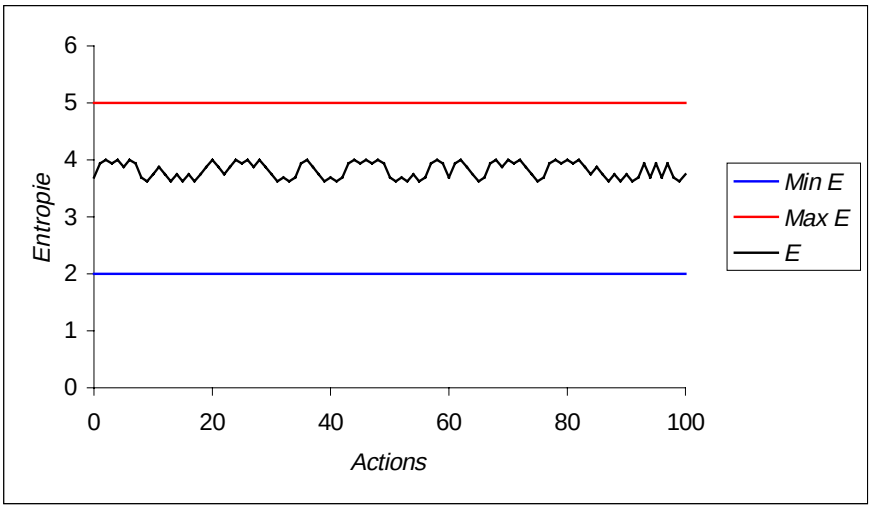
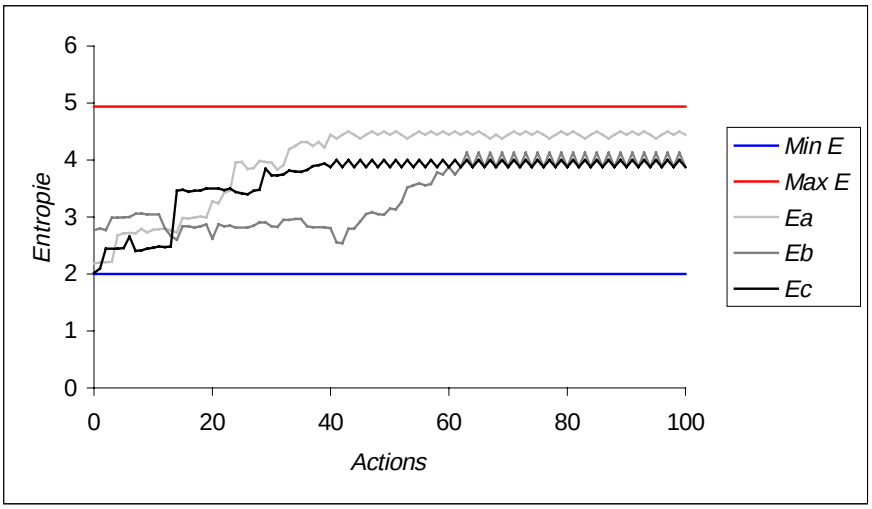


Figure 7 : Oscillation complexe.



Graphique 5 : Oscillation complexe.

Le graphique 6 montre l'évolution, guidée par l'algorithme « local aggregated », de trois SMAM de taille réelle 31. L'évolution est qualitativement identique à celle des SMAM de taille réelle 32. La convergence prend plus de temps que l'algorithme global, l'équilibre parfait n'est pas atteint, et l'évolution est caractérisée par des oscillations.



Graphique 6 : Comportement de l'algorithme « local aggregated » pour *TR*(*S*) = 31.

Nous pouvons conclure que, dans un nombre important de cas, cet algorithme ne simule pas l'évolution naturelle des SMAM.

L'algorithme « global E-matcher II »

Les défauts des algorithmes « global aggregated » et « local aggregated » nous ont motivé à développer des algorithmes plus intelligents qui manipulent non seulement des agents atomique d'ordre supérieur, mais également des agents moins équilibrés.

Chaque algorithme implémentant le comportement « qui se ressemble, s'assemble » doit utiliser un mécanisme pour évaluer la ressemblance entre agents. Ceci correspond, dans le monde des agents, aux fonctionnalités de perception et d'apparence.

Les agents atomiques d'ordre supérieur peuvent, dans les ordinateurs, être représentés d'une manière très compacte : un bit pour indiquer leur statut d'agent atomique d'ordre supérieur et $\log_2(N_{max})$ bits pour représenter leur ordre jusqu'à une valeur maximale de N_{max} - 1. Six bits, par exemple, nous permettent de représenter des agents atomiques d'ordre supérieur ayant une taille réelle inférieure ou égale à $2^{63} \approx 10^{19}$.

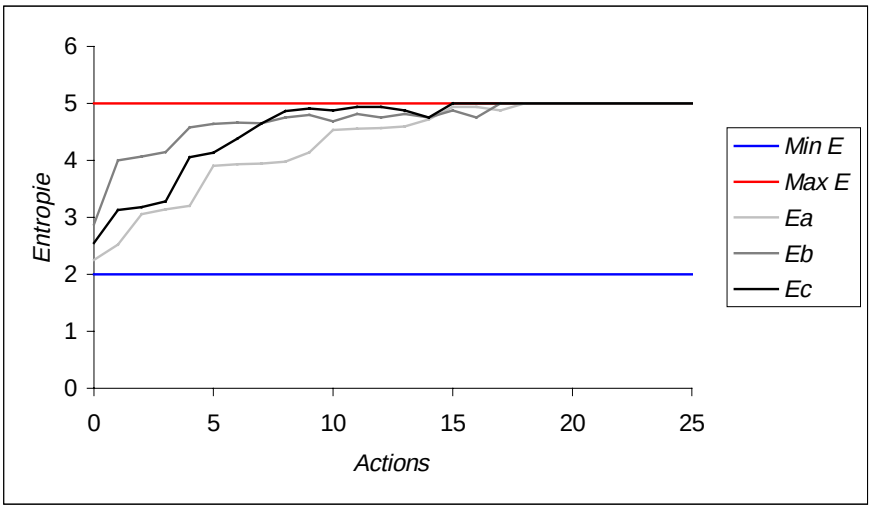
Représenter les autres SMAM d'une manière compacte n'est pas toujours possible et certainement pas simple. En général, nous sommes obligés d'utiliser une abstraction des SMAM pour les comparer. Une comparaison complète nécessiterait une quantité excessive de ressources. Les deux algorithmes « global E-matcher II » et « local E-matcher » utilisent, comme abstraction d'un agent *S*, son entropie E(*S*) qui est une indication de la taille et de l'équilibre de *S*.

L'algorithme est défini (pour *une* action) comme suit :

1) Créer une liste de tous les agents couplés à des partenaires ayant une entropie différente. La liste est ordonnée selon la différence d'entropie entre les agents et leurs partenaires. Les agents les moins bien couplés sont au début de la liste. La taille des agents est utilisée comme critère secondaire de tri.
2) Si la liste est vide, abandonner.
3) Trouver, dans la liste, les deux agents ayant l'écart d'entropie minimal. Les agents vers le début de la liste ont priorité. Le couple des deux doit être nouveau et légal : un agent ne peut pas être couplé à lui-même, ni à son partenaire actuel, ni à son super-agent actuel, ni à un agent faisant partie de lui-même.
4) S'il n'y a pas de couple trouvé, abandonner.
5) Faire migrer le premier agent du couple sélectionné vers le deuxième.

Le graphique 7 nous montre l'évolution, guidée par cet algorithme, de trois SMAM de taille réelle 32. La convergence est rapide et mène les SMAM vers l'équilibre maximal.

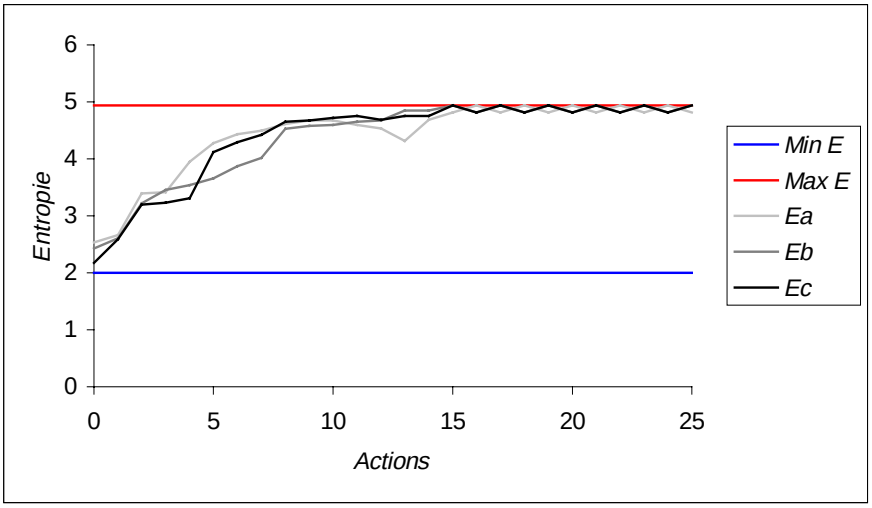
Notons que l'évolution de l'entropie n'est pas monotone comme c'était le cas de l'algorithme « global aggregated ». Toutefois, l'évolution est tout à fait compatible avec l'évolution naturelle des SMAM.



Graphique 7 : Comportement de l'algorithme « global E-matcher II » pour *TR*(*S*) = 32.

Le graphique 8 montre l'évolution de trois SMAM de taille 31, toujours en utilisant l'algorithme « global E-matcher II ». La convergence, jusqu'à une certaine limite, est toujours aussi rapide. Toutefois, une fois l'équilibre maximal atteint, le système commence à osciller entre l'état optimal et un état presque optimal. L'explication de ce phénomène est simple : même dans l'état optimal, certains agents ne sont toujours pas couplés à des agents identiques et continuent à migrer. L'algorithme est conçu de telle manière que ces oscillations ne génèrent que des petites fluctuations d'entropie.

Donc, il s'agit d'un algorithme fiable qui implémente, dans un contexte pratique, correctement l'évolution naturelle des SMAM.



Graphique 8 : Comportement de l'algorithme « global E-matcher II » pour *TR*(*S*) = 31.

L'algorithme « local E-matcher »

Nous avons également construit une version distribuée de l'algorithme « global E-matcher II ».

L'algorithme est défini (pour *une* action) comme suit :

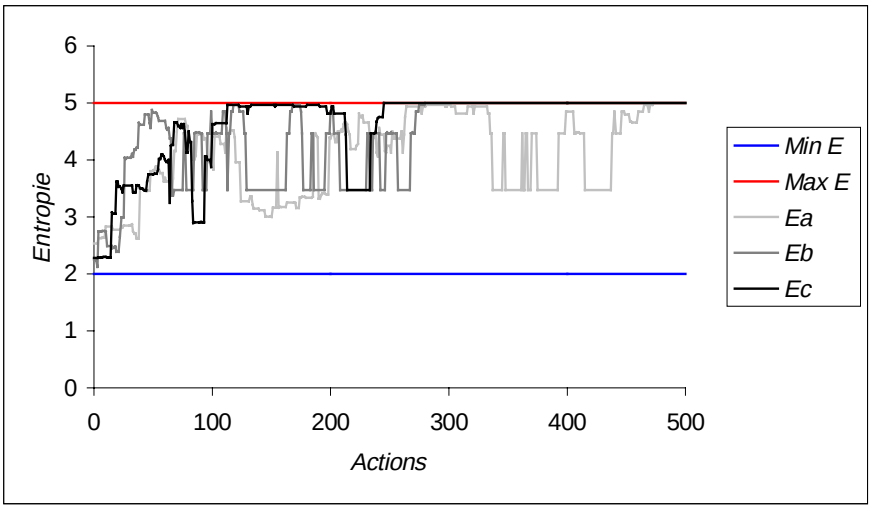
1) Créer une liste de tous les agents couplés à des partenaires ayant une entropie différente.
2) Si la liste est vide, abandonner.
3) Sélectionner, dans la liste, aléatoirement un candidat. Ceci simule le parallélisme.
4) Calculer la différence d'entropie entre le candidat et son partenaire *X* (ΔX), le sous-agent de son partenaire *A* (s'il existe) (ΔA), l'autre sous-agent de son partenaire *B* (s'il existe) (ΔB) et le partenaire de son super-agent (s'il existe) *C* (ΔC).
4) Si $\Delta X > \Delta A$, ou $\Delta X > \Delta B$, ou $\Delta X > \Delta C$, sélectionner *A*, *B* ,ou *C* respectivement comme deuxième candidat. Sinon, abandonner.
5) Faire migrer le premier candidat vers le deuxième.

Le graphique 9 nous montre l'évolution, guidée par cet algorithme, de trois SMAM de taille réelle 32. La convergence est lente, mais elle mène les SMAM vers l'équilibre maximal.

Notons que l'évolution de l'entropie est plutôt irrégulière. Ceci est dû au fait que les migrations ne sont pas planifiées dans le temps d'une manière systématique. En effet, la nature distribuée de l'algorithme ne permet pas, de manière évidente, une telle planification.

Pour que deux agents identiques se rencontrent rapidement dans un SMAM, suivant l'algorithme décrit ci-dessus, ils doivent *avoir de la chance*. Prenons, par exemple, le SMAM de la figure 8 : il n'est pas garanti que *A* et *B* se rejoignent dans un temps donné Δt. La seule chose que l'on peut démontrer, c'est que la probabilité qu'ils

ne se trouvent *pas* diminue avec le temps. Toutefois, ceci suffit pour que l'algorithme implémente l'évolution naturelle des SMAM.



Graphique 9 : Comportement de l'algorithme « local E-matcher » pour *TR*(*S*) = 32.

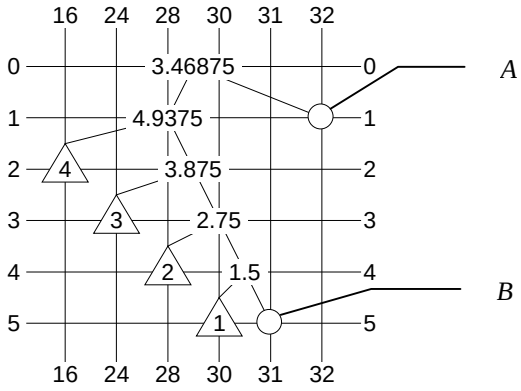
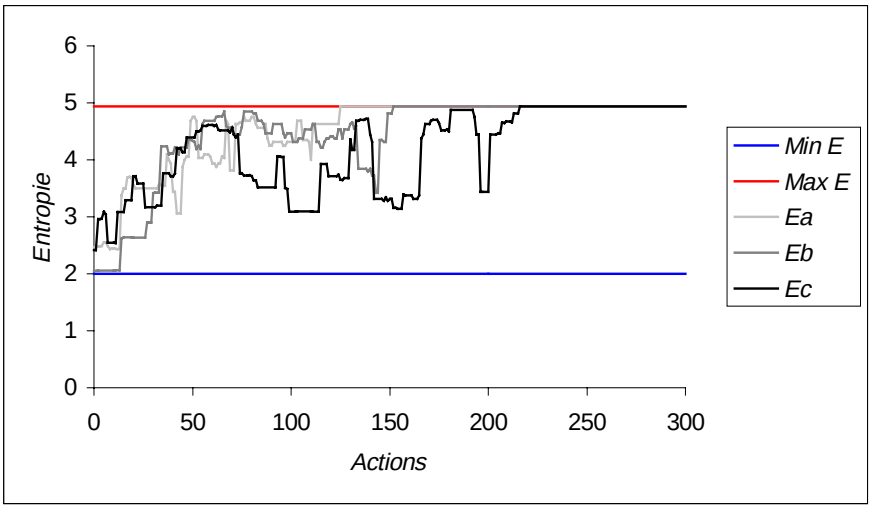


Figure 8 : A la recherche de l'autre.



Graphique 10 : Comportement de l'algorithme « local E-matcher » pour *TR*(*S*) = 31.

Le graphique 10 montre l'évolution, guidée par l'algorithme « local E-matcher », de trois SMAM de taille réelle 31. L'évolution est qualitativement identique à celle des SMAM de taille réelle 32, sauf que la convergence est sensiblement plus rapide.

En expérimentant, nous avons observé que l'algorithme peut nécessiter un temps considérable pour conclure la réalisation d'un agent atomique d'ordre supérieur (comme, par exemple, le cas du SMAM de la figure 8). Toutefois, la convergence est garantie, et l'algorithme simule fidèlement l'évolution des SMAM.

Comparaison des algorithmes

Nous pouvons, en utilisant les critères que nous avons défini au début de ce chapitre, comparer les quatre algorithmes présentés. Leur caractéristiques sont compilées dans le tableau 1 :

Critère / **Algorithme**	**Centralisé / Distribué**	**Simplicité conceptuelle**	**Simplicité algorithmique**	**Maximisation de l'entropie**	**Robustesse**	**Vitesse de convergence**	**Absence d'oscillations**	**Localité des ressources**
« global aggregated »	C	+	0	0	—	+	+	—
« local aggregated »	D	+	+	—	+	0	—	+
« global E-matcher II »	C	+	0	+	+	+	0	—
« local E-matcher »	D	+	+	+	0	—	+	+

Tableau 1 : Comparaison des algorithmes.

A part l'algorithme « local aggregated », qui doit être exclu parce que - dans un grand nombre de cas - il ne simule pas les SMAM, le choix d'un algorithme dépend de l'importance relative des critères. Toutefois, l'algorithme « global aggregated » doit être exclu si la taille réelle du SMAM n'est pas une puissance de deux.

En général, nous utilisons, dans un contexte centralisé, l'algorithme « global E-matcher II » et, dans un contexte distribué, l'algorithme « local E-matcher ».

II.3.3 Un *benchmark* pour les agents cognitifs

Les quatre algorithmes que nous avons présentés sont d'une grande simplicité et servent surtout à illustrer le concept. Dans un contexte plus réaliste, nous ne pouvons souvent utiliser que les algorithmes distribués. Dans ces cas, l'efficacité limitée des algorithmes distribués présentés peut être un grand handicap.

Un système composé de millions d'agents, par exemple, doit être distribué pour des raisons de performance. Un seul ordinateur n'est pas capable de simuler les actions de tous les agents dans des délais acceptables. Un système ouvert, par contre, doit être distribué pour des raisons conceptuelles. En effet, par définition, un système ouvert doit être capable d'accueillir des agents effectuant d'autres algorithmes. Il est donc impossible d'utiliser un algorithme centralisé.

Pour créer des algorithmes distribués plus efficaces, il nous semble très utile de prendre le point de vue « agent », puisque le problème présenté semble être lié de près à certains problèmes étudiés dans les domaines des Agents Autonomes et des Systèmes Multi-Agents.

Un problème pertinent

Nous étudierons la notion d'environnement dans les SMAM en détail dans le chapitre II.5, mais notons déjà que l'environnement d'un agent faisant partie d'un SMAM est composé des autres agents du SMAM. Donc, l'environnement est composé d'agents actifs ayant tous le comportement « qui se ressemble, s'assemble ».

Ceci implique que, si le champ de perception des agents est local, ce qui est toujours le cas dans un contexte réaliste, leur environnement est incertain et dynamique. En effet, à cause des migrations continues, n'importe quel plan de l'environnement est rapidement daté. Les chercheurs dans le domaine des Agent Autonomes travaillent spécifiquement avec ce type d'environnement.

Si les agents du SMAM suivent tous le même algorithme, i.e. s'ils sont **homogènes**, les agents individuels peuvent essayer d'anticiper les action des autres. Ce type de raisonnement sur les autres constitue un sujet important de la recherche dans le domaine des Systèmes Multi-Agents. Par contre, si les agents sont **hétérogènes** et suivent des algorithmes différents, ce type d'anticipation n'est plus possible. Dans ce cas, les agents peuvent tenter d'apprendre ou de comprendre le comportement des autres, dans le but de mieux anticiper les changements dans l'environnement. Ce type d'apprentissage est également un autre sujet étudié dans le domaine des Systèmes Multi-Agents.

Le problème à résoudre, i.e. la maximisation de l'équilibre d'un SMAM, se définit très facilement et sans ambiguïté. Toutefois, il semble très difficile à résoudre d'une manière distribuée, si les champs de perception et de migration sont locaux. D'un point de vue « agent », il implique plusieurs problèmes importants étudiés dans les domaines des Agents Autonomes et des Systèmes Multi-Agents. Ces trois raisons nous ont motivé à proposer le problème de maximisation de l'équilibre d'un SMAM comme un *benchmark* pour des Agents Autonomes dans un contexte Multi-Agents.

Définition du benchmark

Dans un premier temps, nous limitons notre *benchmark* à des SMAM homogènes. Ce problème semble être suffisamment dur. De plus, un *benchmark* pour des SMAM hétérogènes est plus difficile à définir : il nécessite que nous définissions une population d'agents de types différents. Notons que le *benchmark* pour des agents homogènes peut nous aider à construire un catalogue de types d'agents nécessaire pour définir le *benchmark* pour les agents hétérogènes.

Notre benchmark attribue, à chaque type d'agent, une valeur qui indique le temps moyen nécessaire pour un SMAM aléatoire, composé d'agents de ce type et d'une taille donné, à évoluer vers un état d'équilibre maximal.

Le temps peut être exprimé en actions ou, pour prendre en compte la complexité algorithmique des agents, en cycles sur une architecture donnée.

Pour que le *benchmark* soit utile, les limites des champs de perception, de migration et de communication des agents doivent être définies. Ces champs contiennent, respectivement, les agents qu'un agent peut percevoir, les agents vers lesquels un agent peut migrer et les agents avec lesquels un agent peut communiquer.

Le mécanisme de sérialisation, spécifiant comment les actions sont distribuées sur l'ensemble des agents, doit également être défini.

Un exemple de *benchmark* concret est le *benchmark* $B(X, N)$ défini comme suit :

> $S_\alpha(n)$ est un SMAM choisi aléatoirement parmi l'ensemble de tous les SMAM de taille réelle n.
>
> $f_X(S)$ indique le temps nécessaire pour un SMAM S composé d'agents de type X de converger vers un état d'équilibre maximal. Les agents suivent tous le comportement « qui se ressemble, s'assemble ». Leur manière de réaliser ce comportement est dépendante du type X. Toutefois, le champ de perception d'un agent est limité à son partenaire et ses sous-agents, son super-agent et le super-agent et le partenaire de son super-agent. Le champ de migration et le champ de communication sont identiques au champ de perception. Le temps est exprimé en actions. Les actions sont sérialisées dans le temps et distribuées aléatoirement sur l'ensemble des agents.
>
> L'efficacité du type X, selon le *benchmark* $B(X, N)$ est exprimé comme suit :
>
> $$B(X,N) = \sum_{n=2^{N-1}+1}^{2^N} f_X(S_\alpha(n))$$
>
> Le *benchmark* n'est pas défini pour $N = 0$.

$B(X, N)$ nous permet d'évaluer l'efficacité d'un type d'agent donné pour un ordre de taille donné. Par exemple, pour $N = 5$, le *benchmark* teste la performance en utilisant des SMAM de taille 17 jusqu'à 32. Notons, comme le montre le tableau 2, que le benchmark devient trop lourd pour des grandes valeurs de N. D'autres types de *benchmark*, moins simples, mais plus pratiques, peuvent être développés.

Reste la question de savoir comment on peut interpréter la notion d'efficacité. Par exemple, est-ce qu'elle est directement liée à la notion d'intelligence ? Cette question est de nature philosophique et nous ne l'aborderons pas dans ce chapitre dédié aux algorithmes.

N	$TR(S)_{min}$	$TR(S)_{max}$	*# tests*
1	2	2	1
2	3	4	2
3	5	8	4
4	9	16	8
5	17	32	16
6	33	64	32
7	65	128	64
8	129	256	128
9	257	512	256
10	513	1024	512
11	1025	2048	1024
12	2049	4096	2048
13	4097	8192	4096
14	8193	16384	8192
15	16385	32768	16384
16	32769	65536	32768
17	65537	131072	65536
18	131073	262144	131072
19	262145	524288	262144
20	524289	1048576	524288

Tableau 2 : Nombre de tests exigés par $B(X, N)$.

II.3.4 Les SMAM hétérogènes

Un SMAM hétérogène est composé de plusieurs types d'agents suivant tous le comportement « qui se ressemble, s'assemble », mais l'implémentant de manières différentes. Les SMAM hétérogènes nous permettent d'étudier des problèmes liés aux systèmes ouverts permettant l'immigration et l'émigration libre des agents.

Les agents peuvent être de types différents, mais doivent respecter le modèle des SMAM, c'est-à-dire la structure binaire et le comportement à long terme. De plus, dans un contexte réelle, ils doivent respecter les protocoles de perception, de migration et de communication. Ces protocoles doivent être normalisés pour qu'un système hétérogène puisse fonctionner. Evidemment, au-dessus de ces protocoles de base, les agents peuvent établir des protocoles plus élaborés.

A un niveau plus abstrait, nous aimerions savoir s'il est possible de représenter les algorithmes exécutés par les agents comme des SMAM eux-mêmes. Dans ce cas, l'évolution naturelle des SMAM, selon un algorithme universel, correspond à l'exécution des algorithmes. Dans ce cas également, l'algorithme et son exécution sont définis à un niveau supérieur perçu par un observateur. Illustrons le concept à l'aide de la métaphore d'une calculatrice électromécanique : les composants bougent et agissent en suivant des lois physiques, il s'agit d'un processus physique « ordinaire ». Toutefois, du point de vue de l'observateur, l'ensemble exécute un algorithme. Nous reprendrons cette problématique, très liée au concept d'émergence, dans le chapitre II.5, lorsque nous situons notre modèle par rapport à la Vie Artificielle.

En tout cas, la recherche de l'algorithme « mère » et des SMAM représentant des algorithmes ne nous semble pas être un tâche facile. Il est probable que la représentation des algorithmes, même simples, puisse nécessiter des SMAM de très grande taille. Toutefois, une fois cette problématique mieux comprise, nous pourrons exploiter cette compréhension pour construire et analyser des SMAM dont le comportement évolue dans le temps. Une telle connaissance ouvrirait des perspectives fascinantes.

CHAPITRE II.4

REFLEXIONS SUR LE MODELE

If there is a way to do it better, find it!
Thomas A. Edison

II.4.1 Evolution du modèle

Les premières intuitions sur les SMAM sont, d'une façon implicite, exprimées dans *The Art of Insanity* [Van Aeken 94]. Ce pamphlet philosophique, écrit d'une manière intuitive et - par choix - non scientifique, a - en ce qui concerne notre thèse - une valeur historique, plutôt que scientifique.

Nous avons commencé le développement explicite des SMAM en 1995. La première publication qui mentionne le concept est l'article « When Agents Talk - How to Maintain Integrity », publié en 1996 [Van Aeken 96c]. A ce stade, l'essentiel du modèle était établi. Toutefois, la théorie était peu développée et exprimée d'une manière peu optimale.

En 1996, 1997 et 1998, nous avons élaboré la théorie des SMAM et optimisé sa terminologie. En 1998, le modèle a été présenté dans trois articles [Van Aeken 98a] [Van Aeken 98b] [Van Aeken 98c].

En cours de la rédaction de cette thèse, nous avons effectué un petit nombre de changements et d'ajouts à la théorie. La plupart de ces changements sont cosmétiques. Les ajouts consistent surtout en des outils théoriques appliqués à la démonstration de propositions. Toutefois, une modification très importante est l'introduction de la notion de SMAM fermé et le changement conséquent de la définition de l'évolution de la taille d'un SMAM. Cette modification était essentielle pour garantir l'intégrité scientifique du modèle.

Nous avons sincèrement le sentiment que le modèle, comme il est présenté dans ce mémoire, est sans défauts majeurs. En conséquence, nous proposons ce mémoire comme référence actuelle du modèle des SMAM.

En général, les modèles scientifiques évoluent dans le temps. En modélisant les idées scientifiques comme des *memes* [Dawkins 76], et les *memes* comme des SMAM (selon une méthodologie actuellement inconnue), cette évolution peut - en théorie - être vue comme une instance de l'évolution naturelle des SMAM. Dans ce sens, notre modèle peut être utilisé pour analyser sa propre évolution et ses défauts dans un contexte évolutif.

Pour l'instant, il est dur de prédire l'avenir de notre modèle. Récemment, nous avons dû introduire une modification, et il n'est pas exclu que nous serons forcés d'en effectuer une autre demain. Toutefois, le modèle actuel nous semble tellement simple et universel que nous avons confiance en sa stabilité. Cependant, nous avons besoin de meilleurs modèles de l'évolution scientifique pour la garantir. Nous espérons que notre modèle lui-même peut, un jour, être utile dans la prédiction de sa stabilité dans le temps.

II.4.2 Le couple : ensemble ou séquence ?

La représentation lexicale d'un SMAM *S* composé d'un agent *A* et d'un agent *B* n'est ni {*A*, *B*}, ni (*A*, *B*), mais (*A B*). Nous avons choisi cette représentation, parce qu'il ne s'agit ni d'un ensemble de taille libre, ni d'une séquence de deux éléments. Nous avons déjà présenté nos arguments détaillant notre choix qu'un agent composé compte exactement deux agents compositeurs, à savoir notre hypothèse qu'une interaction est toujours entre deux parties et notre volonté de concevoir un modèle minimal. Par contre, nous n'avons pas encore expliqué pourquoi nous traitons les deux agents d'un agent composé comme un ensemble, et non comme une séquence. Nous présentons nos arguments dans cette section.

Dans le monde physique, tout ensemble de deux objets immobiles observés à un temps *t* est ordonné dans l'espace. Si leurs coordonnées sont identiques, les objets doivent être identiques aussi. Une fois un cadre de référence choisi, le premier objet est toujours plus à droite et/ou plus haut et/ou plus loin que le deuxième, sauf quand ils sont identiques.

Au sein d'un ordinateur numérique, tout ensemble de deux données est ordonné dans l'espace mémoire. Lorsqu'on déclare, dans un programme informatique, une structure de données, même s'il s'agit d'un ensemble dans le sens mathématique, on établit, explicitement ou implicitement, leur ordre.

Apparemment, la notion d'ensemble mathématique est une *abstraction* qui n'est pas nécessairement réalisable dans le vrai monde. La question de savoir si l'ensemble mathématique *existe* est certainement très intéressante, mais elle nous mène dans des territoires dangereux. A cause de notre éducation, nous prenons son concept pour évident. Mais, comme nous le montre l'histoire des mathématiques, il ne l'est pas. Par exemple, une des crises les plus importantes de ce domaine, autour du début du siècle, était liée au paradoxe de Bertrand Russell. Ce paradoxe, découvert par Russell pendant sa rédaction de *Principles of Mathematics* [Russell 03] et discuté en détail dans une annexe de cet ouvrage, est le suivant : prenons l'ensemble $\Omega = \{ S \mid S \notin S \}$, est-ce que $\Omega \in \Omega$? Si $\Omega \in \Omega$, $\Omega \notin \Omega$ et nous avons une contradiction. Si $\Omega \notin \Omega$, $\Omega \in \Omega$ et nous avons aussi une contradiction. Puisque, dans la logique classique, n'importe quelle proposition peut être démontrée à partir d'une contradiction, un grand nombre de mathématiciens étaient plutôt perturbés par cette découverte.

Plusieurs solutions ont été apportées. Russell lui-même a proposé la *théorie des types* [Russell 08]. A la base de cette théorie se trouve la construction d'une hiérarchie classifiant les objets mathématiques et éliminant le paradoxe. David Hilbert et les formalistes ont suggéré de ne considérer que les objets finis, bien définis et constructibles et des règles d'inférence certaines. Luitzen Brouwer et les intuitionists ont proposé que les seuls objets mathématiques existants sont ceux dont on sait démontrer comment les construire. Une quatrième solution, apporté par Ernst Zermelo consiste en la restriction de la manière dont les ensembles mathématiques peuvent être définis [paradoxe].

Les trois premières solutions au paradoxe de Russell introduisent des notions de construction et, dans un certain sens, d'algorithme dans la théorie de l'ensemble mathématique. Elles rendent sa notion moins abstraite. La quatrième solution, par contre, respecte plus sa nature abstraite. Ceci peut bien être la raison pour laquelle elle a été retenue, dans une forme légèrement modifiée par Abraham Fraenkel, par la plupart des mathématiciens aujourd'hui.

L'histoire du paradoxe de Russell illustre bien le fait que la notion de l'ensemble mathématique est moins évidente qu'elle semble. En effet, depuis un temps respectable, tout un domaine scientifique est dédié à la théorie des ensembles. Reste la question de savoir pourquoi nous, dans notre théorie, utilisons la notion de l'ensemble mathématique, sachant qu'elle implique certaines complications. La réponse est simple : parce qu'elle est la plus élégante.

Les SMAM ont la tendance naturelle à évoluer vers un état d'équilibre maximal. Dans le cas idéal (d'un point de vue « esthétique ») où le système est fermé et sa taille réelle est une puissance de deux, le système évolue vers un état dans lequel les agents atomiques se trouvent tous dans le même contexte et dans lequel tous les agents atomiques sont échangeables : la symétrie est parfaite. Ceci peut être visualisé mentalement en regardant l'exemple de la figure 1 et en imaginant que les structures représentant les agents de chaque couple tournent rapidement autour de l'axe formé par le disque représentant l'agent composé.

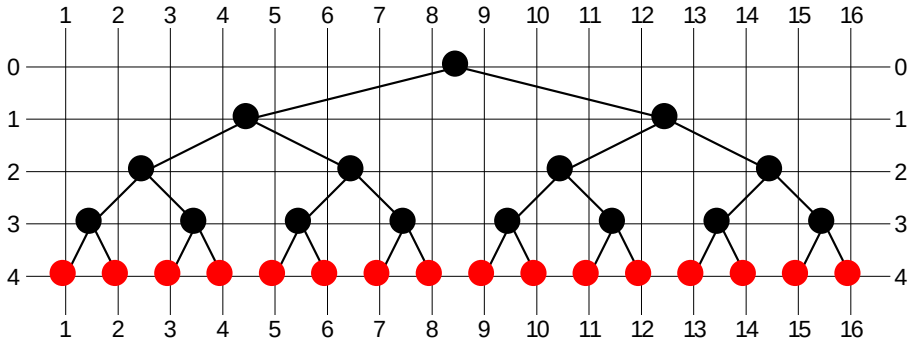


Figure 1 : Symétrie parfaite.

Maintenant, imaginons que les deux agents d'un couple ne sont pas échangeables, qu'il existe une position 'gauche' et une position 'droit' différente. Dans ce cas, aucun des agents atomiques du SMAM de la figure 1 peut changer de position : ils sont ordonnés de gauche à droite, de 1 à 16. Nous avons perdu notre symétrie parfaite…

Le même argument peut être présenté au niveau microscopique. De notre point de vue, le couple idéal ne distingue pas entre ses deux membres, mais exprime une égalité totale.

Nous devons, très probablement, payer un prix pour notre choix. Au début de cette section, nous avons indiqué qu'il est dur, peut-être impossible, de trouver des instances de l'ensemble mathématique dans le monde réel. Pire, du point de vue de l'informaticien pur et dur, nous ne savons pas implémenter des vrais ensembles. Donc, nous ne saurons peut-être jamais construire des vrais Systèmes Multi-Agents Minimaux, seulement des simulations.

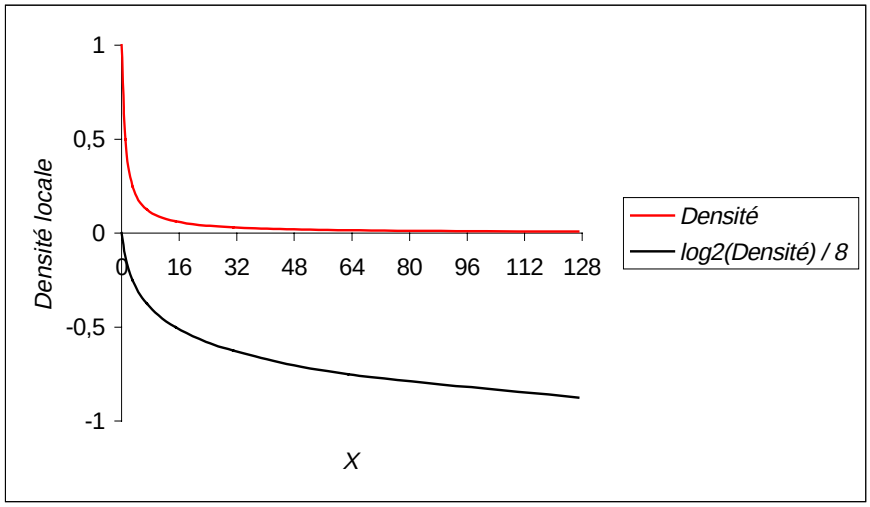
Nous avons fait notre choix. Toutefois, la notion du SMAM « sexué », distinguant entre les deux sous-agents des agents composés, mène à des questions intéressantes. Par exemple, lorsqu'un agent A migre vers un agent B et un couple est formé, qui sera l'agent 'gauche' et qui sera l'agent 'droit' ? Autrement dit, quelle est la sémantique de la polarité du couple ? Plusieurs réponses sont possibles. Toutefois, dans un premier temps, concentrons-nous sur les SMAM comme nous les avons définis et posons des questions plus pertinentes. Par exemple, pourquoi le comportement des agents est basé sur l'attraction et non sur la répulsion entre agents ?

II.4.3 Attraction ou répulsion ?

Le mécanisme « qui se ressemble, s'assemble » implique une force d'attraction entre des agents similaires. Toutefois, dans le monde physique, nous trouvons non seulement des forces attractives, mais également des forces répulsives entres des éléments du même type. Par exemple, les deux pôles 'nord' de deux aimants se repoussent. De la même façon, deux électrons se repoussent. Est-ce que notre modèle est incomplet pour modéliser de tels phénomènes, sommes nous obligés d'ajouter un deuxième comportement de base à nos agents, ou même de modifier le comportement de base ? Nous croyons que ce n'est pas nécessaire et qu'il suffit d'attaquer la modélisation de ces phénomènes par le bon angle.

Illustrons cette approche en considérant un ensemble de particules distribuées, d'une manière inégale, sur un segment de l'espace en une dimension. Notons que nous ne modélisons que le comportement répulsif. Il ne s'agit pas d'un modèle de particules physiques comme elle sont étudiées dans la physique. Un tel modèle implique nécessairement une complexité beaucoup plus importante.

Imaginons que nous avons 2^N particules distribuées d'une manière très déséquilibrée (plus qu'exponentielle) sur le segment. Le graphique 1 montre la densité locale des particules en fonction de leur position :



Graphique 1 : Distribution de particules.

A cause du comportement apparemment répulsif des particules, le système évoluera vers un état de densité constante sur tout le segment. Comment est-ce qu'on peut expliquer cette évolution en termes de SMAM ?

La clé à la solution de cette question se trouve en réalisant que le système étudié consiste non seulement en 2^N particules, mais également en un segment de l'espace euclidien en une dimension. Si nous voulons modéliser le système, nous sommes obligés de modéliser le segment aussi.

Si nous modélisons les particules par des agents atomiques, nous pouvons modéliser le segment en associant le segment complet au SMAM complet et en attribuant une bissection récursive du segment à chaque agent composé. La figure 2 illustre ce principe pour un simple SMAM.

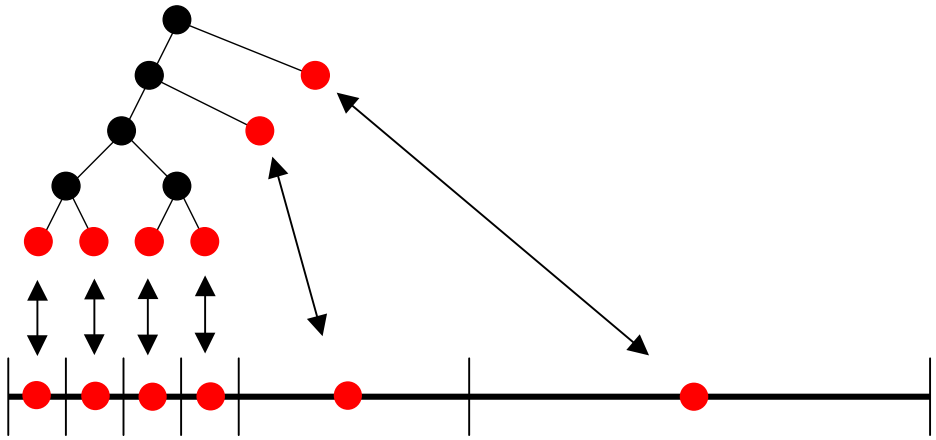
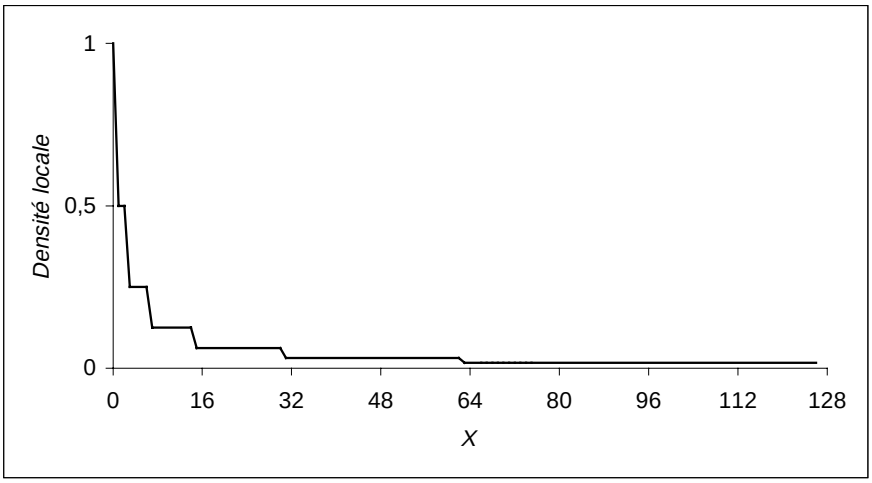


Figure 2 : Correspondance entre particules et agents atomiques.

Notons que cette approche ne mène pas à une modélisation complète du segment de l'espace euclidien en une dimension. Par exemple, le SMAM ne définit pas l'ordre des sous-segments identifiés. Toutefois, pour le but de notre démonstration, le modèle proposé suffit.

Si nous prenons $N = 7$, la situation du graphique 1 correspond à un SMAM linéaire d'une taille réelle de 128. En utilisant notre modèle de particules et espace, la densité locale des particules est celle illustrée par le graphique ci-dessous.



Graphique 2 : Distribution de particules (modèle SMAM).

En suivant, au niveau des agents atomiques, le principe « qui se ressemble, s'assemble », le SMAM évoluera naturellement vers un SMAM atomique d'ordre 7 modélisant une distribution égale des particules. Toutefois, du point de vue de l'observateur, les particules ont suivi le principe « qui se ressemble, se repousse ».

En général, on a tendance à prendre l'espace comme une donnée évidente, constante et neutre, mais peut-être qu'elle ne l'est pas. En tout cas, comme nous l'avons montré, sa modélisation explicite peut nous aider à décrire certains phénomènes intéressants en termes de SMAM.

Notons que l'espace, dans le modèle présenté dans cette section, est délimité par son continu, à savoir les particules. Là où il n'y a plus de particules, il n'y a plus d'espace. Autrement dit, dans ce modèle, le vide absolu n'existe pas. Puisque le modèle n'implique que des densités extrêmement petites aux limites de l'espace, cela nous semble pas être un problème inquiétant.

II.4.4 « Les extrêmes s'attirent » ?

Dans la section précédente, nous avons suggéré comment modéliser, dans un cas spécifique mais important, le comportement « qui se ressemble, se repousse » en termes de SMAM. Dans cette section, nous étudions la question symétrique, à savoir : comment peut-on marier le concept des SMAM au comportement « les extrêmes s'attirent » ?

Nous pouvons reprendre l'exemple des deux aimants. Cette fois, le phénomène intéressant est le fait que le côté 'nord' de l'un attire le côté 'sud' de l'autre. D'une manière similaire, les électrons attirent les protons. Ou encore, dans un contexte biologique, les mâles et les femelles s'attirent. Est-ce que ces phénomènes ne contredisent pas tous le comportement « qui se ressemble, s'assemble » ?

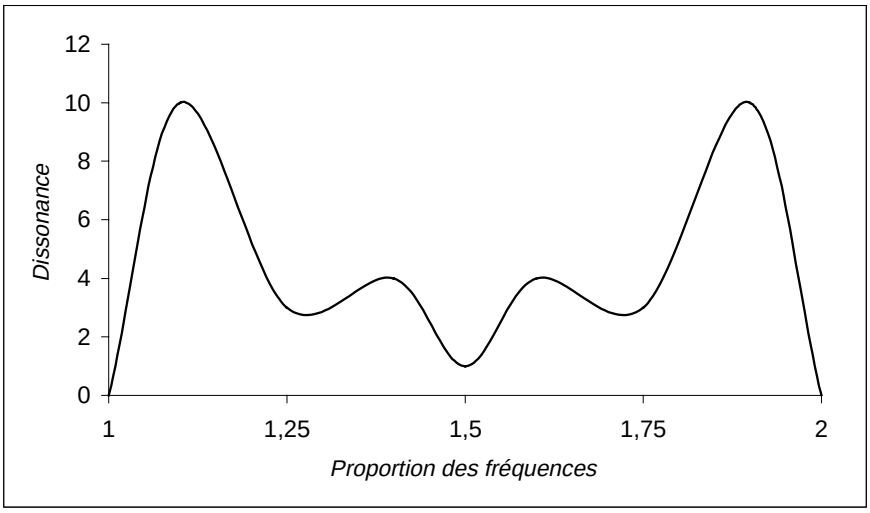
Il s'agit d'une question difficile et nous essayons d'y répondre en deux temps. D'abord, nous montrons que les « extrêmes » en général sont souvent très similaires. Puis, nous toucherons à la problématique de la polarisation des agents et ses conséquences.

Le pôle 'nord' d'un aimant n'attire pas n'importe quoi. Il n'est attiré que par des pôles magnétiques (ayant la qualité d'être un pôle 'sud'). Les électrons aussi ne sont attirés que par des particules assez similaires. Le même argument s'applique au troisième exemple qui est peut-être le plus parlant : les mâles d'une espèce ne sont pas attirés par les femelles de n'importe quelle espèce, seulement par celles de la *même* espèce.

Ces exemples montrent que la qualification d'« extrême » est, jusqu'à un certain degré, synonyme à « similaire, mais non identique ». Une des meilleures illustrations de ce phénomène peut être trouvée dans le monde de la musique. Dans nombre d'expériences, des chercheurs dans ce domaine ont présenté, à un groupe d'auditeurs, plusieurs combinaisons de deux sons et leur ont demandé d'évaluer le degré de dissonance de la combinaison. De chaque combinaison, les deux sons sont de fréquences stables, mais - en général - différentes [Taylor 94] [Roederer 94] [Howard 96]. Le graphique 3 donne une bonne impression des résultats typiques de ces expériences.

Dans ce graphique, l'axe X représente la proportion des fréquences des deux sons. L'axe Y représente le degré de dissonance comme elle est aperçue par les auditeurs. La dissonance atteint une valeur maximale lorsqu'on approche la consonance parfaite. Jusqu'à une certaine limite, les sons les plus proches semblent être les sons les plus incompatibles.

Une conséquence importante de ce phénomène, dans le contexte de notre discours, est que, durant l'évolution d'un système vers l'équilibre parfait, des comportements apparemment contradictoires au comportement « qui se ressemble, s'assemble » peuvent être observés. La « dissonance » observée peut augmenter et on peut avoir l'impression que « les extrêmes s'attirent ». Toutefois, comme nous l'avons illustré, ces phénomènes peuvent être secondaires et dépendants de l'observateur.



Graphique 3 : Dissonance de deux sons.

Hors du rôle de l'observateur, le mécanisme « les extrêmes s'attirent » s'appuie presque toujours sur une polarisation des agents et les comportements complexes qui en résultent. Dans un contexte biologique, il s'agit de la question du *sexe*. Une question qui est loin d'être résolue, comme l'expriment avec éloquence Charles Taylor et David Jefferson [Taylor 95] :

> « All else being equal, a female who reproduces asexually will leave twice as many genes in her offspring as will a female who reproduces with a male. This would seem to impose a tremendous hurdle for sexuality to overcome, yet it persists, and sex is widespread in the natural world. Why? The answer almost certainly involves complex interactions among linkage, pleiotropy, epistasis, parasitism, and nonlinear relations between genotype and fitness. »

Pour que la polarisation des agents soit persistante dans le contexte des SMAM, les agents eux-mêmes doivent maintenir cette polarisation, dans le même sens qu'un organisme biologique doit maintenir sa forme générale (voir le chapitre II.5). Ceci implique que les agents polarisés doivent être des SMAM de grande taille et de grande complexité. Nous supposons que le comportement associé, « les mâles et les femelles s'attirent », soit d'une complexité comparable et qu'il ne s'agisse pas d'un comportement de base. En conséquence, comme dans le cas biologique, une explication satisfaisante du phénomène peut être loin d'être triviale. Sa recherche est, toutefois, fascinante et nous espérons étudier, un jour, ce phénomène dans le monde des SMAM.

II.4.5 Visualiser les SMAM

Les Systèmes Multi-Agents Minimaux sont des systèmes abstraits et donc, par définition, non visibles. Evidemment, pour communiquer le concept, mais aussi pour nous aider à le manipuler dans notre esprit et pour mieux le comprendre, nous sommes obligés de définir une représentation visuelle des SMAM. Au cours du chapitre II.2, nous avons déjà présenté plusieurs représentations visuelles. Dans cette section, nous les parcourons systématiquement et les classifions selon un nombre de critères différents. Nous introduisons également une nouvelle représentation importante inspirée par les arbres « H ».

Critères de classification

Tout un domaine scientifique est dédié aux techniques de dessin de graphes. Une bonne introduction à ce domaine se trouve à [graph drawing]. La bibliographie annotée par Di Battista et al [Di Battista 94] constitue une excellente source de références dans le domaine, même si elle ne couvre que des travaux jusqu'à 1994. L'étude, nécessairement superficielle, du domaine nous a aidé à situer et à classifier nos méthodes de visualisation.

Nous pouvons grouper les représentations visuelles des SMAM dans deux classes essentielles : **lexicale** et **graphique**. Une représentation lexicale s'appuie sur un alphabet spécifique pour représenter un SMAM comme une séquence de caractères. Une représentation graphique utilise des primitives graphiques tel que des disques ou des triangles.

Les représentations graphiques se décomposent en deux sous-classes : **connecté** et **non connecté**. Dans une représentation connectée, les entités représentant les agents d'un couple sont graphiquement connectées à l'entité représentant le couple. Ceci n'est pas le cas dans une représentation non connectée. Les représentations connectées ressemblent de plus près aux représentations classiques des graphes.

Une représentation lexicale n'exploite qu'une dimension spatiale. Les représentations graphiques utilisent, en général, deux dimensions. On peut également envisager des représentations en trois dimensions ou plus. Le **nombre de dimensions utilisées** est une caractéristique importante.

Certaines représentations n'utilisent qu'une couleur. D'autres sont polychromes. Les représentations peuvent être classifiées selon le **nombre de couleurs utilisées**.

Evidemment, le critère le plus important est la **facilité d'interprétation** de la représentation. Plus spécifiquement, la représentation idéale nous donne non seulement une **bonne vue de l'ensemble**, mais également une **bonne vue des détails**. Aussi, certaines représentations **expriment bien la récursivité** des SMAM. D'autres **expriment bien le groupement** des SMAM.

Un caractéristique très importante est la possibilité d'**ajouter des étiquettes** aux agents. Ces étiquettes peuvent, par exemple, indiquer l'entropie des agents.

Dans certains contextes, la **facilité de manipulation** des agents dans la représentation peut être un critère de grande valeur.

Certaines représentations permettent d'**exprimer**, dans un sens très limité, **la dynamique** des SMAM. Cette capacité peut être utile pour représenter les SMAM associés à un nombre de modèles d'organisation classiques.

Important aussi, dans certains cas, est la **qualité esthétique** d'une représentation. Evidemment, ce critère est lié à des préférences personnelles et culturelles. Toutefois, notons que, dans la littérature anglo-saxonne, la qualité esthétique est considérée plus ou moins comme synonyme de la lisibilité. Puisque nous avons identifié d'autres critères directement liés à la lisibilité, nous utiliserons la qualité esthétique comme un critère indépendant des critères fonctionnels. Son évaluation est nécessairement personnelle.

L'**insensibilité à l'effet de crénelage** (*aliasing*) peut beaucoup différer d'une représentation à l'autre. D'un point de vue graphique et esthétique, ceci peut être un critère important.

La **légèreté du dessin** constitue le dernier critère que nous avons pris en considération. Elle est liée au temps de calcul nécessaire pour générer la représentation.

Les représentations actuelles

A l'heure actuelle, nous avons à notre disposition cinq représentations différentes : la représentation originale, lexicale, arborescente, triangulaire, et « arbre H ». Dans ce qui suit, nous les présentons en visualisant l'agent atomique, l'agent composé de deux agents atomiques et l'agent de la figure 3. Aussi, nous les analysons en utilisant les critères identifiés précédemment. A la fin de la section, nous présentons un tableau comparatif des cinq représentations.

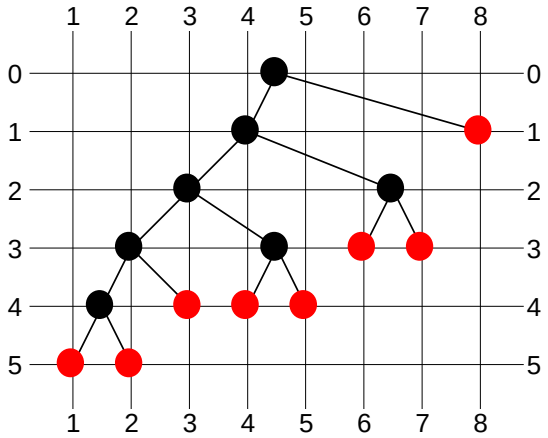


Figure 3 : SMAM « modèle ».

La représentation originale

Tout au début du développement du modèle, nous avons utilisé une représentation visuelle qui exprimait bien les intuitions sans être parfaite. Elle est utilisée dans la publication « When Agents Talk » [Van Aeken 96c]. En général, elle a été remplacée par les autres représentations.

L'agent atomique est présenté comme un disque noir :



Figure 4 : Représentation originale (agent atomique).

L’agent composé de deux agents atomiques se présente sous forme d’un carré aux angles arrondis :

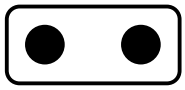


Figure 5 : Représentation originale (agent composé).

Si l’agent composé est un agent atomique d’ordre supérieur, son style de trait est épais (comme dans la figure 3). Dans tous les autres cas, son style de trait est fin, comme le montre la figure 6 qui visualise le SMAM de la figure 3.

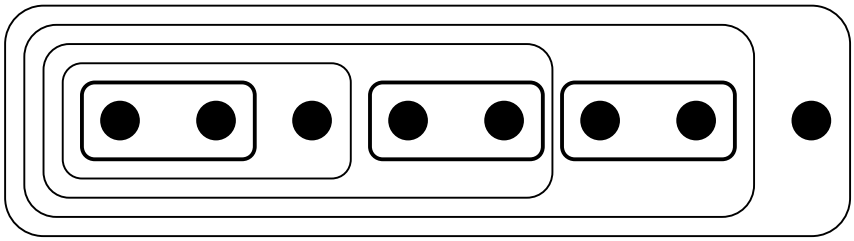


Figure 6 : Représentation originale (agent de la figure 3).

La distance entre deux disques noirs adjacents est constante pour un dessin donné.

Comme dans toutes nos représentations, l’agent *G* d’un couple (*G D*) est visualisé à gauche si sa taille est égale ou supérieure à la taille de *D*. Si *G* est différent de *D* mais toutefois de la même taille, le choix est libre.

Notons que, en utilisant la terminologie de la littérature sur le dessin de graphes, la représentation originale suit la convention « inclusion ».

Comme l’illustre la figure 7, l’arbre binaire correspondant au SMAM peut, si on le souhaite, être visualisé aussi.

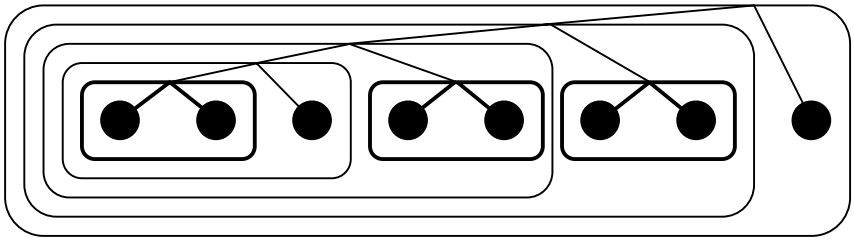


Figure 7 : Représentation originale, visualisant l’arbre correspondant.

La représentation de l’arbre ajoute au dessin des traits droits connectant les nœuds (*straight-line polyline drawing*). Aussi, elle est planaire (*planar drawing*), c’est à dire aucun trait ne coupe un autre. Les traits connectant les sous-agents d’un agent atomique d’ordre supérieur sont épais comparés aux autres traits.

La représentation originale est intéressante d’un point de vue historique et peut être utile dans un contexte didactique. Toutefois, elle souffre de certains défauts. En particulier, elle n’offre pas une grande lisibilité. Ses caractéristiques principales sont compilées dans le tableau 1 qui se trouve vers la fin de cette section.

La représentation lexicale

Cette représentation, simple et formelle, est basée sur l'utilisation de quatre symboles (caractères), à savoir 'Δ', '(', ')' et ' '.

L'agent atomique est présenté comme : Δ

Si, dans un contexte typographique donné, le caractère 'Δ' n'est pas disponible, il peut exceptionnellement être remplacé par le caractère 'A'.

Un agent composé de deux agents atomiques est présenté comme : (Δ Δ)

Notons que la représentation des deux sous-agents d'un couple n'est pas séparée par une virgule mais par une espace, et ceci pour trois raisons :

— l'utilisation d'une virgule n'aide pas à la lisibilité,
— l'utilisation d'une virgule complique le dessin, et
— l'utilisation d'une virgule évoque l'image du couple mathématique.

Aussi, nous avons opté plutôt pour les caractères '(' et ')', que pour les caractères '{' et '}', et ceci pour des raisons très similaires :

— l'utilisation d'accolades n'aide pas à la lisibilité,
— l'utilisation d'accolades complique le dessin, et
— l'utilisation d'accolades évoque l'image de l'ensemble mathématique.

Comme toujours, l'agent *G* d'un couple (*G D*) est écrit à gauche si sa taille est égale ou supérieure à la taille de *D*. Si *G* est différent de *D* mais toutefois de la même taille, le choix est libre.

En utilisant la représentation lexicale, l'agent de la figure 1 est présenté comme : (((((Δ Δ) Δ) (Δ Δ)) (Δ Δ)) Δ)

Les grands avantages de la représentation lexicale sont sa compacité, sa facilité de dessin, et son universalité : elle peut être reproduite sur n'importe quelle machine à écrire. Sa lisibilité, par contre, est très limitée. Nous avons compilé ses caractéristiques principales dans le tableau 1.

La représentation arborescente

La représentation arborescente est la plus utilisée des cinq représentations existantes. Aussi, c'est celle qui ressemble le plus à la représentation classique des arbres binaires.

Cette méthode de visualisation exige l'utilisation de deux couleurs différentes, à savoir rouge et noir. Dans des contextes monochromes, la couleur rouge peut exceptionnellement être remplacée par la couleur gris.

L'agent atomique est présenté comme un disque rouge :



Figure 8 : Représentation arborescente (agent atomique).

L'agent composé de deux agents atomiques se présente, plus ou moins, sous forme de la représentation classique d'un arbre binaire : un dessin composé d'un disque noir représentant, *comme une entité séparée*, l'agent composé, et donc la totalité du SMAM, deux disques rouges représentant les agents atomiques et deux traits droits connectant les nœuds. Notons que le dessin est planaire : aucun trait ne coupe un autre.

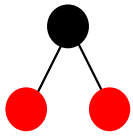


Figure 9 : Représentation arborescente (agent composé).

En utilisant cette représentation, le SMAM de la figure 3 se visualise comme suit :

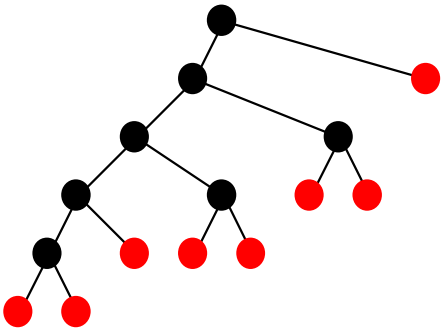


Figure 10 : Représentation arborescente (agent de la figure 3).

Dans un dessin donné, la distance (selon l'axe X) entre deux agents atomiques est constante. Aussi, dans un dessin donné, la distance (selon l'axe Y) entre deux agents à des niveaux adjacents est constante. Un agent composé est toujours dessiné au centre (selon l'axe X) de l'ensemble des agents atomiques qui le constituent.

Comme dans toutes nos représentations, l'agent *G* d'un couple (*G D*) est visualisé à gauche si sa taille est égale ou supérieure à la taille de *D*. Si *G* est différent de *D* mais toutefois de la même taille, le choix est libre.

On peut, si on le souhaite, ajouter une grille qui indique les niveaux des agents et les rangs des agents atomiques. Ceci est illustré par la figure 11.

Dans une variante de cette représentation, des étiquettes représentant les valeurs d'entropie des agents sont ajoutées au dessin. Dans ce cas, l'agent atomique est représenté comme un disque vide suggérant sa valeur d'entropie de 0. Les disques noirs représentant les agents composés sont remplacés par les étiquettes correspondantes. En utilisant cette représentation, et en affichant la grille, l'agent de la figure 3 se présente comme dans la figure 12.

Dans encore une autre variante, plus compacte, les agents atomiques d'ordre supérieur différent de 0 sont visualisés comme des triangles rouges sur lesquels les valeurs de l'ordre sont superposées. En utilisant cette représentation, toujours en affichant la grille, l'agent de la figure 3 est visualisé comme dans la figure 13.

Notons que, dans cette représentation, la grille indique les niveaux des agents et les rangs *accumulés* des agents atomiques.

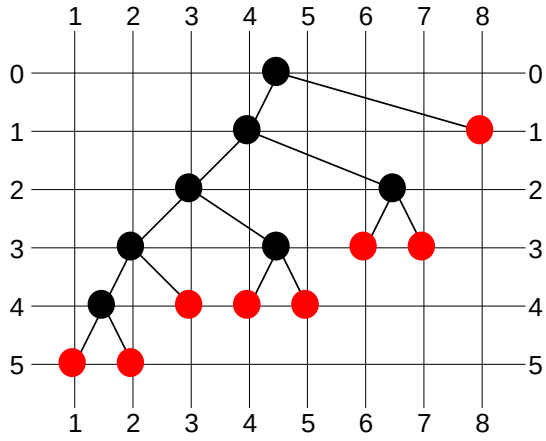


Figure 11 : Représentation arborescente, visualisant la grille.

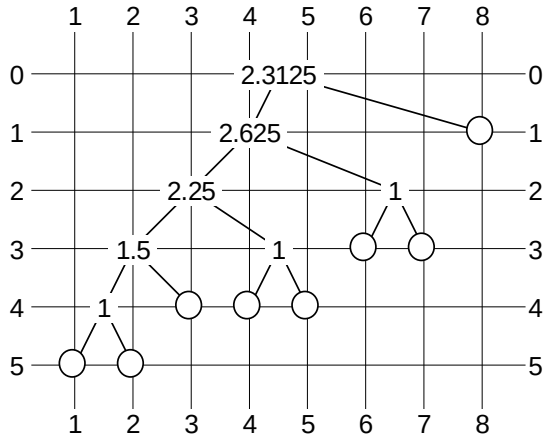


Figure 12 : Représentation arborescente, visualisant les valeurs d'entropie.

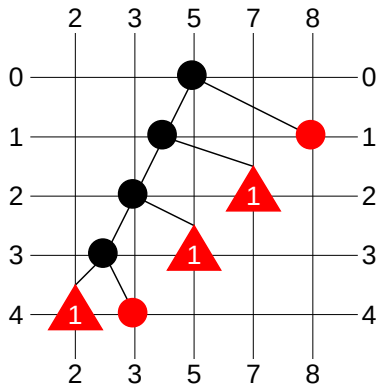


Figure 13 : Représentation arborescente, compacte.

De cette variante de la représentation arborescente existe également une version affichant les valeurs d'entropie des agents. Dans cette représentation, les disques et les triangles représentant les agents atomiques d'ordre supérieur ne sont pas remplis et les disques noirs représentant les agents composés sont remplacés par les étiquettes correspondantes. Puisque l'entropie d'un agent atomique d'ordre supérieur est égale à son ordre, il

n'est pas nécessaire d'ajouter des étiquettes aux représentations de ces agents. La figure 14 montre la représentation de l'agent de la figure 3, toujours en affichant la grille.

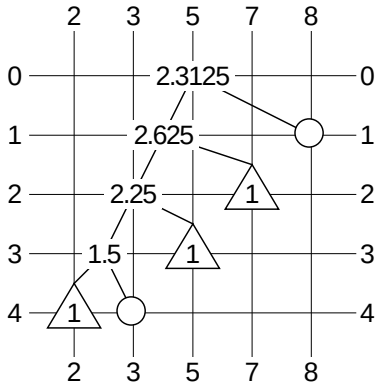


Figure 14 : Représentation arborescente, compacte, visualisant les valeurs d'entropie.

La représentation arborescente nous permet d'exprimer, d'une manière très spécifique, la dynamique d'un SMAM. Si les sous-agents d'un nombre donné d'agents composés migrent entre eux d'une manière continue, nous avons l'option d'identifier ces agents composés à l'aide d'une courbe fermée. Cette option est compatible avec les autres options de la représentation arborescente. La figure 15 montre la représentation d'un SMAM dont les agents *A*, *B* et *C* migrent continuellement entre eux.

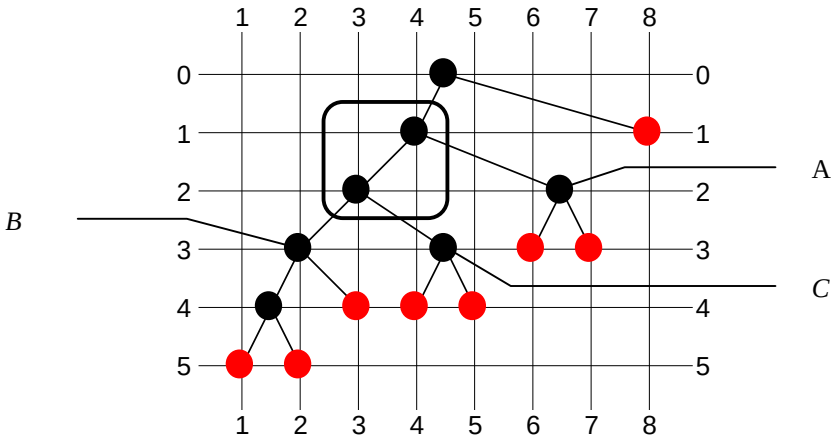


Figure 15 : Représentation arborescente, migration continue.

La représentation arborescente s'est avérée très utile dans nos expériences du modèle des SMAM, parce qu'elle permet une analyse et une manipulation aisée des agents. Pour ces raisons, elle est au centre de notre plate-forme d'expérimentation FRIENDS Offline. Par contre, elle n'est pas très intuitive, ce qui limite ses qualités didactiques. L'essentiel de ses caractéristiques principales est compilé dans le tableau 1.

La représentation triangulaire

La représentation triangulaire est sans doute la représentation la plus intuitive. Elle exprime le mieux la nature récursive et le groupement des agents.

Elle est le résultat de notre recherche de la forme géométrique la plus simple, en deux dimensions, qui puisse facilement être coupée en deux morceaux ayant la même forme. Ceci pour exprimer, sans concessions, la récursivité et le groupement des agents.

Les formes les plus simples, en deux dimensions, sont les triangles et les cercles. Il est clair qu'un cercle ne peut pas être coupé en deux cercles. Est-ce qu'il existe un type de triangle dont on puisse couper les instances en deux triangles du même type ? En effet, un tel type existe. Il suffit de prendre un triangle dont un angle est de 90° et dont les deux autres sont de 45° et de le couper entre les points **A** et **B** :

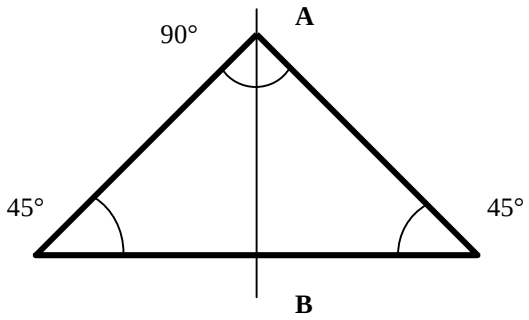


Figure 16 : Triangle à la base de la représentation triangulaire.

Dans la représentation triangulaire, l'agent atomique est visualisé comme un triangle (90°, 45°, 45°) dont la couleur est rouge. La figure 17 montre un agent atomique ainsi visualisé. Dans des contextes monochromes, la couleur rouge peut exceptionnellement être remplacée par la couleur gris.

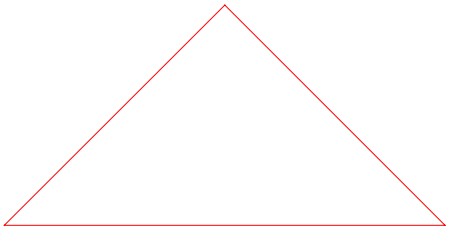


Figure 17 : Représentation triangulaire (agent atomique).

Comme le montre la figure 18, l'agent composé de deux agents atomiques est visualisé comme un triangle (90°, 45°, 45°) noir contenant les représentations des agents atomiques. Si l'espace le permet, une bordure de taille libre est laissée autour des agents composants.

Notons que l'orientation de la représentation des deux agents atomiques est différente de l'orientation de la représentation de l'agent composé. Au total, huit orientations différentes existent. Notons que, comme la représentation originale, la représentation triangulaire suit la convention « inclusion ».

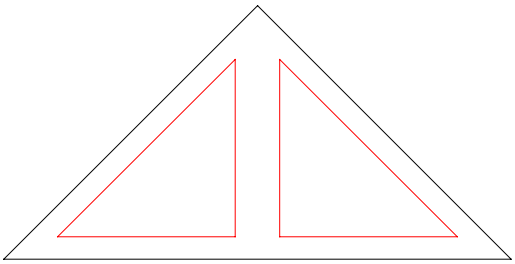


Figure 18 : Représentation triangulaire (agent composé).

En utilisant la représentation triangulaire, l'agent de la figure 3 se visualise comme suit :

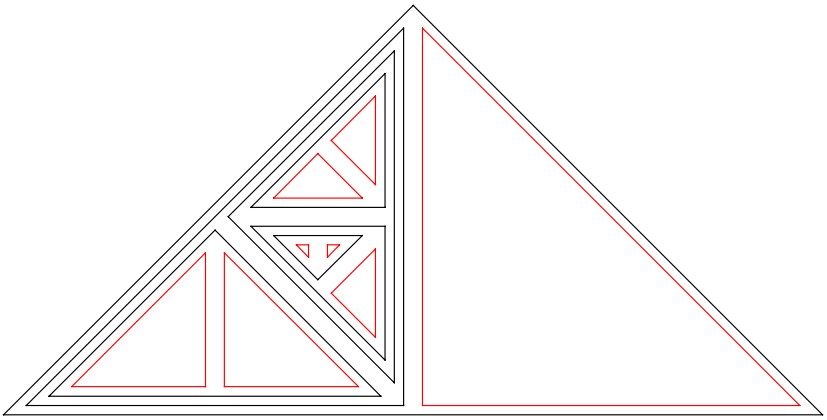


Figure 19 : Représentation triangulaire (agent de la figure 3).

L'agent *G* d'un couple (*G D*) est visualisé à gauche, relatif à la base du triangle, si sa taille est égale ou supérieure à la taille de *D*. Si *G* est différent de *D* mais toutefois de la même taille, le choix est libre.

Dans un dessin donné, la taille des bordures autour des triangles est constante, ce qui donne un effet plus esthétique que lorsqu'on utilise des bordures de tailles variables. En conséquence, à partir d'un certain niveau, il n'y a plus de place pour dessiner les bordures. A partir de ce niveau, les agents composés ne sont plus dessinés. La figure 20 montre la représentation du SMAM de la figure 19, mais en utilisant des bordures d'une taille supérieure.

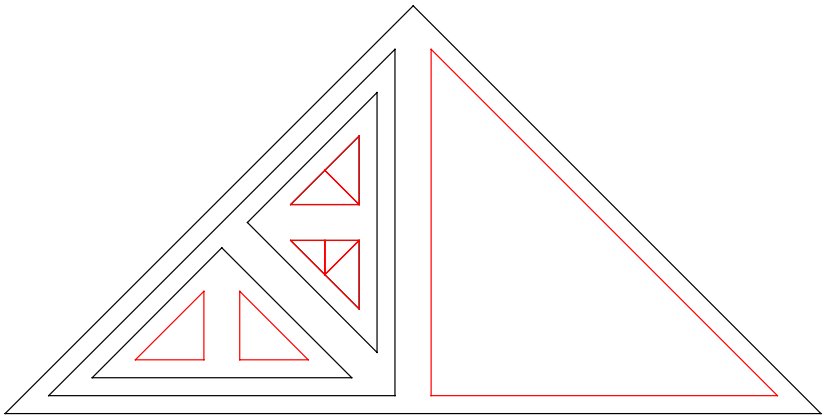


Figure 20 : Représentation triangulaire, bordures larges.

Comme la représentation arborescente, la représentation triangulaire nous permet d'exprimer, d'une manière limitée, la dynamique d'un SMAM. Si les sous-agents d'un certain nombre d'agents composés migrent continuellement entre eux, nous avons l'option d'identifier ces agents composés en utilisant un trait plus gras. La figure 21 montre la représentation d'un SMAM dont les agents *A*, *B* et *C* migrent continuellement entre eux.

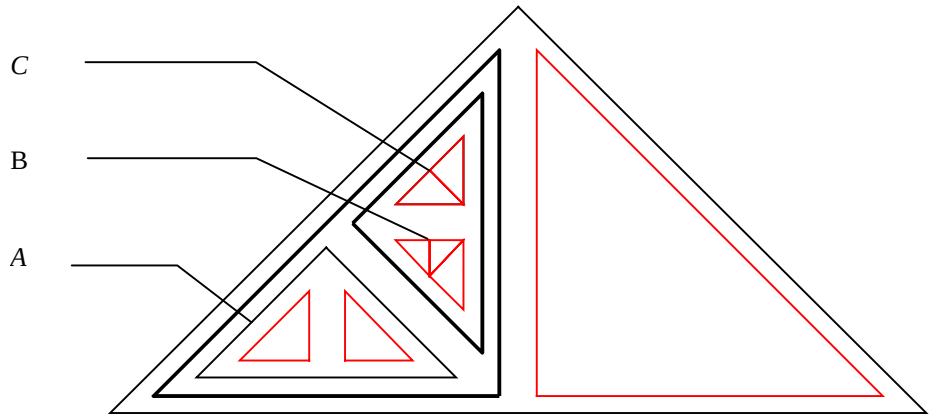


Figure 21 : Représentation triangulaire, migration continue.

La représentation triangulaire est très intuitive et donne une très bonne vue de l'ensemble d'un SMAM. Par contre, à partir d'un certain niveau, les détails ne sont plus visibles. Aussi, il est difficile de manipuler les agents ou d'y attribuer des étiquettes. Notons également que la représentation est très sensible à l'effet de crénelage : une erreur légère dans le calcul des coordonnées peut engendrer des artefacts peu esthétiques dans le dessin. L'ensemble des caractéristiques de la représentation triangulaire peut être consulté dans le tableau 1.

Notons que, dans des outils informatiques, le défaut principal de la représentation triangulaire, à savoir son impuissance à bien visualiser les détails, peut être surmonté en ajoutant une fonction « zoom » permettant de descendre et de remonter dans les niveaux du SMAM.

La représentation « arbre H »

La dernière méthode de visualisation que nous présentons est inspirée par la notion d'arbres H. Les arbres H sont une représentation graphique en deux dimensions d'arbres comptant jusqu'à quatre fils par nœud.

Dans la représentation « arbre H », l'agent atomique est visualisé comme un segment rouge (gris dans un contexte monochrome). La figure 22 montre un agent atomique ainsi visualisé.

Figure 22 : Représentation « arbre H » (agent atomique).

La figure 23 montre la visualisation d'un agent composé de deux agents atomiques. L'agent composé, comme entité isolée, est représenté comme un segment noir. Notons que les trois traits constituant le dessin ont tous une orientation différente. Au total, il existe quatre types d'orientation.

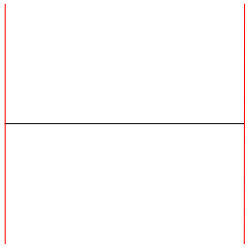


Figure 23 : Représentation « arbre H » (agent composé).

Figure 24 nous montre la représentation « arbre H » du SMAM de la figure 3.

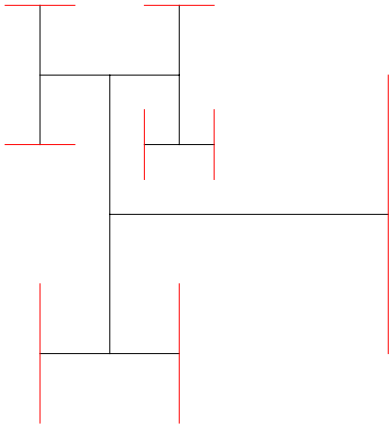


Figure 24 : Représentation « arbre H » (agent de la figure 3).

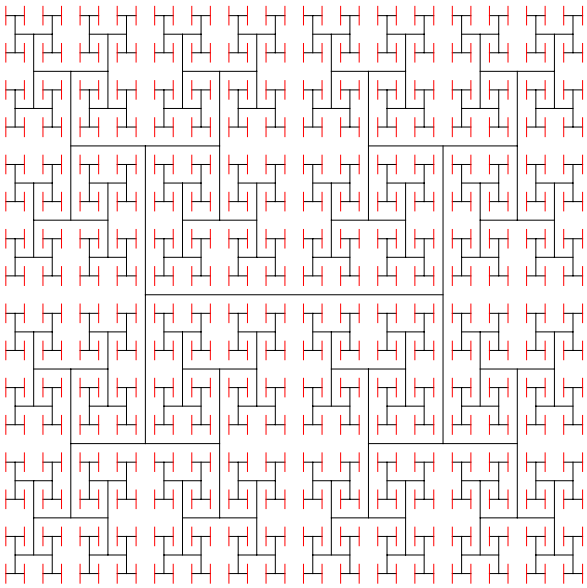


Figure 25 : Représentation « arbre H » d'un agent atomique d'ordre 8.

La représentation « arbre H » exploite, mieux que les autres représentations que nous avons définies, l'espace en deux dimensions. En conséquence, elle nous offre non seulement une bonne vue de la totalité d'un SMAM, mais aussi de ses détails. Ceci est illustré par la figure 25, montrant un agent atomique d'ordre 8, comptant plus de 500 agents. Le défaut majeur de la représentation est le fait qu'elle n'exprime pas du tout le groupement des agents. Nous avons compilé l'ensemble de ses caractéristiques dans le tableau 1.

Comparaison des représentations

Nous avons compilé, dans le tableau 1, nos évaluations des cinq représentations présentées, en termes des critères définis au début de la section.

Comme l'indique le tableau, le choix de type de visualisation dépend surtout de son utilisation déterminant l'importance relative des critères.

Critère / **Type de visualisation**	**Lexicale / Graphique**	**Connecté / Non connecté**	**Nombre de dimensions utilisées**	**Nombre de couleurs utilisées**	**Facilité d'interprétation**	**Bonne vue de l'ensemble**	**Bonne vue des détails**	**Exprimant bien la récursivité**	**Exprimant bien le groupement**	**Possibilité d'ajout d'étiquettes**	**Facilité de manipulation**	**Exprimant la dynamique**	**Qualité esthétique**	**Insensibilité à l'effet de crénelage**	**Légèreté du dessin**
Originale	G	(C)	2	1	—	—	+	0	+	—	—	—	0	+	+
Lexicale	L	N	1	1	—	—	—	+	+	—	—	—	0	+	+
Arborescente	G	C	2	2	—	—	+	+	—	+	+	+	0	+	0
Triangulaire	G	N	2	2	+	+	—	+	+	—	—	+	+	—	+
« Arbre H »	G	N	2	2	0	+	+	+	—	—	0	—	+	+	+

Tableau 1 : Comparaison des représentations visuelles.

Le côté esthétique

Les représentations triangulaires et « arbre H » ont, plus que les autres représentations, un côté esthétique fascinant. Chez certaines personnes, ils peuvent évoquer des images de dessins « Maya » ou indien en général. Toutefois, pour une représentation donnée, certains SMAM sont plus appréciés que d'autres. Une analyse scientifique de ce phénomène nous semble très intéressante. Est-ce que, par exemple, les gens préfèrent la « presque perfection », comme c'est souvent le cas dans la musique ?

II.4.6 Les SMAM et les arbres binaires

Il existe une forte ressemblance entre certaines représentations des SMAM et les arbres binaires. En conséquence, on peut se poser la question de savoir si les théories et les algorithmes des arbres binaires, sujet bien étudié dans l'informatique, sont applicables aux SMAM. Dans cette section, nous présentons la notion d'arbre binaire comme elle est utilisée dans l'informatique et la comparons au concept de SMAM.

Définition des arbres binaires

Berlioux & Bizard définissent l'arbre binaire comme suit [Berlioux 88] :

On appelle **arbre binaire** sur un ensemble *D* une arborescence telle que

(a) chaque sommet a zéro, un ou deux fils,
(b) quand un sommet a deux fils, ceux-ci sont ordonnés,
(c) l'ensemble des sommets de l'arborescence est l'ensemble *D*.

Etudions la correspondance entre les SMAM et les arbres binaires de la définition ci-dessus. Notons d'abord qu'un SMAM n'est ni une arborescence, ni un graphe en général, main un Système Multi-Agents composé de plusieurs agents actifs. Un graphe est, par définition, passif et manipulé par un algorithme séparé. Un SMAM est, par définition, actif et ne nécessite pas d'entité externe pour évoluer. Toutefois, on peut comparer les *arborescences correspondantes aux* SMAM aux arbres binaires.

En ce qui concerne (a), notons qu'il est impossible qu'un agent composé n'ait qu'un seul fils. En effet, l'agent composé représente le *couple* de ses deux fils et il ne peut pas exister sans eux. Selon la définition de Berlioux & Bizard, il n'y a pas de différence conceptuelle entre les nœuds non feuilles et les feuilles : une feuille est définie comme un nœud ayant zéro fils. Par contre, dans les SMAM, les agents composés et les agents atomiques sont fondamentalement différents. Donc, pour une classe importante d'arbres binaires, il n'existe pas de SMAM ayant des arborescences correspondantes.

En ce qui concerne (b), signalons que les deux fils d'un agent composé ne sont pas ordonnés. Il n'y a pas d'ordre dans un couple : c'est un ensemble. Nous avons motivé ce choix ultérieurement. Donc, les arborescences correspondantes aux SMAM sont différentes des arbres binaires définies par Berlioux & Bizard.

En ce qui concerne (c), nous avons déjà remarqué qu'il existe une différence importante entre les agents atomiques et les agents composés d'un SMAM : ce sont deux classes différentes. Dans le contexte des arbres binaires, il n'existe pas une telle distinction.

Compilant nos remarques à propos de (a), de (b) et de (c), nous concluons que, selon la définition de Berlioux & Bizard, les arborescences correspondantes aux SMAM ne sont pas des arbres binaires.

Etudions une deuxième définition d'arbres binaires, celle formulée par Cormen, Leiserson et Rivest [Cormen 94] :

Un **arbre binaire** T est une structure définie sur un ensemble fini de nœuds qui :

— ne contient aucun nœud, *ou*

— est formé de trois ensembles de nœuds disjoints : un nœud *racine*, un arbre binaire qu'on appelle son *sous-arbre gauche*, et un arbre binaire appelé *sous-arbre droit*.

Cette définition est équivalente à celle de Berlioux & Bizard. Donc, selon Cormen, Leiserson et Rivest aussi, des différences importantes existent entre les arborescences correspondantes aux SMAM et les arbres binaires.

Les arbres binaires complets

Selon Cormen, Leiserson et Rivest, un arbre binaire est **complet** lorsque chaque nœud est soit une feuille, soit d'un degré exactement égal à 2. Notons qu'une feuille est un nœud sans fils et que le degré d'un nœud est le nombre de fils d'un nœud.

Les arbres binaires complets sont plus proches du concept des SMAM que les arbres binaires en général. Toutefois, des différences importantes subsistent.

Les arbres binaires complets et non ordonnés

Dans la littérature « informatique », les arbres binaires sont typiquement définis comme **ordonnés**, c'est-à-dire il existe une différence entre le fils 'gauche' et le fils 'droit' de chaque nœud : ils ne forment pas d'ensemble dans le sens mathématique. Par contre, dans le domaine des graphes, un arbre binaire n'est pas nécessairement ordonné.

Les arbres binaires complets et non ordonnés ressemblent de près aux arborescences correspondantes aux SMAM. Toutefois, ce type d'arbre est très peu étudié comparativement aux arbres binaires classiques, non complets et ordonnés, définis ci-dessus. En effet, l'arbre binaire le plus étudié dans l'informatique n'est même pas l'arbre binaire classique, mais l'arbre binaire de recherche.

Les arbres binaires de recherche

Lorsqu'on attribue à chaque nœud d'un arbre binaire un élément (une valeur) d'un ensemble V, lorsqu'on défini une relation de trie sur les éléments de V, et lorsqu'on restreint la structure d'un arbre binaire classique d'une manière spécifique, on obtient **un arbre binaire de recherche**.

Il est clair que ce type d'arbre est encore plus éloigné des SMAM que l'arbre binaire classique. Toutefois, pendant les cinquante dernières années, un très grand nombre de recherches ont été effectuées sur les arbres binaires de recherche. Les chercheurs ont développé un grand nombre de sous-types et d'algorithmes correspondants (B-trees, AVL-trees, etc.). Il est donc légitime de se demander si on peut appliquer ces algorithmes aux SMAM.

Comme nous l'avons montré, ceci n'est pas possible sans changer la nature essentielle des SMAM. De plus, un SMAM est beaucoup plus que son arborescence, il s'agit d'un système actif et naturellement distribué, très différent d'une structure de données passive. Donc, malgré la ressemblance superficielle, nous ne pouvons pas directement appliquer aux SMAM les recherches effectuées - dans l'informatique - sur les arbres binaires.

CHAPITRE II.5

SITUER LE MODELE

> *If someone points out to you that your pet theory of the universe is in disagreement with Maxwell's equations - then so much the worse for Maxwell's equations. If it is found to be contradicted by observation - well, these experimentalists do bungle things sometimes. But if your theory is found to be against the second law of thermodynamics I can give you no hope; there is nothing for it but to collapse in deepest humiliation.*
>
> Sir Arthur Eddington

II.5.1 Les SMAM et les Systèmes Multi-Agents

Le modèle au centre de notre thèse est conçu pour modéliser une classe spécifique de Systèmes Multi-Agents. Dans cette section, nous situons ce modèle et la classe qu'il modélise dans le contexte global des SMA. A cette fin, nous décomposons notre modèle selon les quatre dimensions identifiées par Yves Demazeau dans sa méthodologie « voyelles ». Il s'agit spécifiquement des axes « Agents », « Environnement », « Interactions » et « Organisation » [Demazeau 97].

Notons que, dans cette section, nous ne discutons que des SMAM *purs*. Dans virtuellement toutes les applications des SMAM, nous appliquons un modèle légèrement modifié que nous avons baptisé le modèle des SMAM *augmentés*. Nous avons développé cette variante pour des raisons pratiques et non pour des raisons conceptuelles. Les SMAM augmentés se situent essentiellement de la même façon que les SMAM purs dans l'espace de tous les SMA.

Les agents

Dans les SMAM, nous pouvons identifier deux types d'agents différents : les agents atomiques et les agents composés. Les agents atomiques peuvent représenter pratiquement toute entité active et autonome imaginable, pourvu qu'elle suive le comportement « qui se ressemble, s'assemble ». Notre modèle ne spécifie ni l'architecture des agents ni la façon de laquelle ils implémentent leur comportement de base. Donc, les agents peuvent être réactifs ou cognitifs, basés sur un système expert, sur un réseau de neurones artificiels, ou sur toute autre technologie. Comme nous l'avons indiqué dans le chapitre II.3, notre modèle est un modèle conceptuel qui laisse ouvert les dimensions architecturales et algorithmiques.

Les agents composés, par contre, sont restreints par leur définition : ils représentent le couple de deux SMAM. A part cette restriction, ils jouissent de la même liberté que les agents atomiques et seul leur comportement conceptuel est défini.

Nous ne croyons pas qu'il existe, dans le domaine des Systèmes Multi-Agents, d'autres modèles qui exigent le comportement « qui se ressemble, s'assemble » des agents. En général, le comportement des agents, même au

niveau conceptuel, est ouvert et peut être choisi en fonction de l'application ou de l'expérience qu'on veut réaliser. Dans un certain sens, cette différence entre les SMAM et les SMA est comparable à la différence entre les automates cellulaires et le Jeu de la Vie comme il a été formulé par Conway [Gardner 83]. Ces deux modèles sont d'un grand intérêt, même si le comportement des cellules est libre dans le premier et fixe dans le deuxième. Le fait que le comportement des entités de base dans le Jeu de la Vie est fixe ne constitue pas de restriction conceptuelle, car on peut démontrer qu'il est possible de construire un ordinateur universel dans le monde du Jeu de la Vie [Berlekamp 82]. D'une manière similaire, le comportement fixe des agents d'un SMAM n'implique pas nécessairement que le comportement du système soit trivial au niveau global.

Notre modèle est récursif dans le sens qu'un agent composé représente le couple de deux SMAM. La notion de récursivité exprime la similarité entre un objet et ses composants. Plus une entité ressemble à ses constituants, plus sa récursivité est pure. Evidemment, dans des implémentations réelles, la récursivité ne peut pas être infinie : donc, en pratique, elle n'est jamais complètement pure. Guidé par notre souhait de construire un modèle minimal, nous avons essayé de pousser la récursivité au maximum. Par exemple, notre choix du comportement « qui se ressemble, s'assemble » est directement lié à la récursivité du modèle. En effet, peu de comportements d'agents sont adaptés à une structure récursive et gardent leur sémantique à travers un grand nombre de niveaux récursifs. Malgré les limites que la récursivité impose sur les modèles qui la supportent, plusieurs modèles récursifs sont proposés dans la communauté SMA. Toutefois, la récursivité dans ces modèles n'est, souvent, pas très pure.

En général, dans le domaine des SMA, la notion de récursivité implique que les SMA soient considérés comme des agents à un niveau supérieur. Toutefois, les agents qui constituent un SMA dit récursif sont souvent d'un type différent du SMA même. Dans ce cas, la similarité entre les niveaux différents et le nombre même de niveaux sont relativement limités. De tels modèles sont, entre autres, proposés par Serge Stinckwich [Stinckwich 94], Pierre Marcenac [Marcenac 97], Michel Occello [Occello 97b] et Samir Aknine [Aknine 98]. Cette approche particulière offre de nombreux avantages mais la récursivité dans ces modèles est tellement impure que le terme « multi-niveaux » nous semble plus correct pour la décrire.

L'environnement

Dans le contexte des Systèmes Multi-Agents, plusieurs types d'environnements peuvent être distingués. Selon la définition d'Yves Demazeau, l'environnement est un ensemble d'objets externes à la population des agents, mais interne au SMA. Cette notion d'environnement semble être la plus répandue dans le domaine. Selon elle, l'environnement d'un SMAM est vide car un SMAM ne contient que des agents.

Toutefois, nous pouvons considérer chaque agent d'un SMAM comme étant entouré par un environnement composé d'agents. Dans le contexte des SMAM, nous définissons cet **environnement interne** (ou social) d'un agent comme son complément relativement au SMAM global. Plus exactement :

$$\forall S \in \mathbb{S}, A \subseteq \mathrm{S}, ENV_S(A) = \{ X \subseteq S \mid X \nsubseteq A \}$$

Notons que l'environnement interne d'un agent n'est pas un SMAM, mais un ensemble de SMAM.

La figure 1 illustre la notion en montrant un SMAM S, un agent $A \subseteq S$ et $ENV_S(A)$.

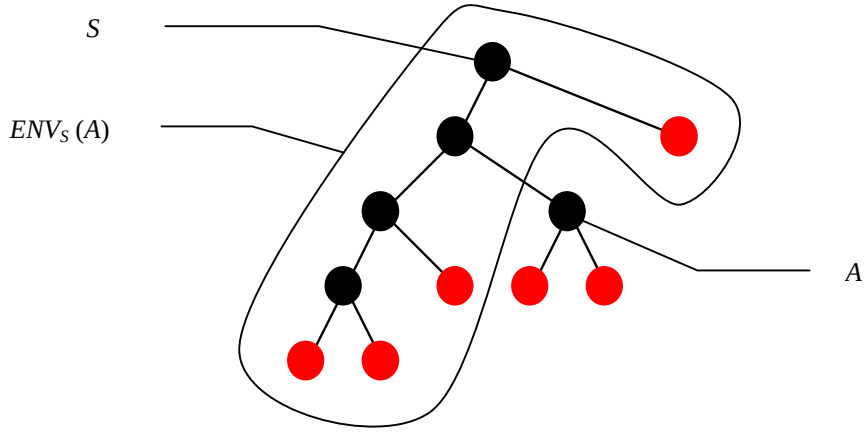


Figure 1 : Environnement interne de l'agent *A*.

Nous pouvons également définir l'**environnement interne *mesuré*** d'un agent, qui est un sous-ensemble de l'environnement interne et qui nécessite la spécification d'un rayon autour de l'agent. Plus formellement :

$$\forall S \in \mathfrak{S}, A \subseteq S, \delta \text{ un nombre naturel, } ENV_{S,\delta}(A) = \{ X \subseteq S \mid X \not\subseteq A \text{ et } d(X, A) \leq \delta \}$$

Lorsque nous parlons de l'environnement *proche* d'un agent d'un SMAM, nous nous référons à son environnement interne mesuré pour un δ relativement petit. L'environnement interne non mesuré peut être vu comme un cas spécial de l'environnement interne mesuré :

$$\forall S \in \mathfrak{S}, A \subseteq S, ENV_S(A) = ENV_{S,\infty}(A)$$

Notons que la notion d'environnement interne mesuré coïncide - jusqu'à un certain degré - à celle de *champ* comme elle est développée dans [Van Aeken 90a] [Van Aeken 90b] [Baeijs 96] et [Ferrand 94].

Dans le monde abstrait des SMAM, l'environnement interne, mesuré ou non, est le seul type d'environnement que nous pouvons identifier. Toutefois, un système concret, construit par l'homme, ne peut pas exister tout seul. S'il s'agit d'un système expérimental, il est observé par des scientifiques. Si, par contre, il s'agit d'une application, le système est manipulé par des utilisateurs. Dans les deux cas, nous pouvons identifier un **environnement externe** au système qui interagit avec lui. Cet environnement peut être différent d'agent à agent. Par exemple, dans les applications que nous avons développées, les agents atomiques représentent des utilisateurs. L'environnement externe de chaque agent atomique consiste principalement en son utilisateur personnel.

Un système réel doit être implémenté au-dessus d'un support concret, par exemple un ordinateur numérique équipé d'un système de gestion. Ces éléments également constituent un environnement dans lequel le système est situé. Nous l'appelons l'**environnement informatique** (ou opérationnel). Dans la décomposition classique selon Marr, l'environnement informatique peut être complètement isolé du système [Marr 82]. Ceci est exact pour des systèmes entièrement abstraits. Par contre, pour des systèmes observés par de scientifiques ou manipulés par des utilisateurs, l'environnement informatique peut jouer un rôle important vis-à-vis du système. En effet, plus l'environnement externe est important, plus l'environnement informatique joue.

Les notions d'environnement externe et d'environnement informatique sont applicables au SMA en général, même si - pour l'instant - peu de chercheurs les modélisent. Le concept d'environnement interne joue toujours un rôle dans les SMA, même s'il n'est pas toujours traité de manière explicite. Par contre, la notion d'un environnement composé d'objet non-agents, populaire dans le domaine, ne fait pas partie de notre modèle. Si on veut intégrer des objets dans notre formalisme, on est obligé de les modéliser comme des agents.

Les interactions

Les interactions de base entre les agents d'un SMAM consistent en des migrations. A un niveau plus cognitif, les agents peuvent communiquer par l'échange de messages. La migration joue un rôle essentiel dans notre modèle, impliquant une organisation dynamique des agents et nécessitant un protocole de migration. Nous l'étudierons plus en détail vers la fin de cette section.

Les agents d'un SMAM peuvent, dans l'implémentation du comportement « qui se ressemble, s'assemble », utiliser la communication symbolique. En échangeant des messages, ils ont la possibilité de communiquer des informations à propos de leur environnement proche, ce qui peut augmenter sensiblement leur efficacité, pourvu que leur champ de perception soit plus petit que leur champ de communication. De manière symétrique, ils peuvent lancer des requêtes du style « Est-ce que quelqu'un me ressemble ? », en spécifiant une abstraction d'eux-mêmes. Ces requêtes peuvent être passées d'agent à agent qui y ajoutent leur coordonnées. Ce mécanisme peut, jusqu'à un certain degré, permettre à un récepteur, ressemblant à l'émetteur, de retrouver l'émetteur de ce message.

Toute forme de communication nécessite un langage ou, au moins, un protocole de communication. Plusieurs langages conçus spécifiquement pour la communication entre agents existent. Parmi les plus connus nous trouvons KQML [Finin 94] et ACL [FIPA]. L'utilité de ces langages est directement liée à l'existence d'une ontologie commune entre agents, ce qui est - particulièrement dans des systèmes ouverts - souvent dur à réaliser. Il s'agit en quelque sorte du problème d'enracinement comme il a été formulé par Stevan Harnad [Harnad 90]. En effet, s'il n'existe pas de racines communes entre agents, il est difficile de réaliser une ontologie partagée. Dans le cas des SMAM, les messages échangés entre agents réfèrent à des concepts intérieurs à leur monde et non à des concepts d'un univers extérieur. Spécifiquement, ils ne parlent que des SMAM et de la migration. Donc, dans un SMAM homogène, le problème d'enracinement ne se pose pas. Dans un SMAM hétérogène, par contre, le problème peut se poser puisque les agents peuvent manipuler des concepts à des niveaux supérieurs qui ne sont pas nécessairement partagés entre les types différents d'agents. Par exemple, un type d'agents peut utiliser le concept de temps (exprimé en actions), complètement ignoré par un autre type. Dans ce cas, les agents doivent utiliser un mécanisme spécifique pour construire une ontologie commune.

D'un point de vue très particulier, les agents migrants peuvent être considérés comme des messages. Il s'agit de la notion de *message actif*. Sous un autre point de vue encore, les migrations peuvent être considérées comme des calculs. Cette notion ouvre des perspectives fascinantes, car elle nous permet de modéliser des systèmes exécutant des algorithmes classiques de l'Intelligence Artificielle comme des systèmes réactifs. A l'heure actuelle, cette piste reste à être explorée.

Indépendamment du point de vue qu'on prend, les interactions constituent un élément essentiel des Systèmes Multi-Agents. Jacques Ferber, pionnier des SMA, les estime tellement importants qu'il a proposé de nommer le

domaine « la kénétique », la science des interactions [Ferber 95]. Dans les SMAM également, les interactions sont à la base de leur évolution et donc de leur nature propre.

L'organisation

Puisqu'un SMAM est identifié par sa structure et puisque sa structure est synonyme de son organisation, ce quatrième axe identifié par Yves Demazeau prend une place spéciale dans cette thèse. Virtuellement tous les aspects des SMAM sont liés directement au concept d'organisation comme nous l'avons défini pour les SMAM.

En situant notre modèle, soulignons que, dans le domaine des SMA, la notion d'organisation est presque toujours liée à des relations de pouvoir entre agents ou à une décomposition de la fonctionnalité globale d'un système en un ensemble de fonctionnalités partielles. En particulier, la notion de **rôle** est directement liée à celle d'organisation. Dans le contexte des systèmes réactifs ou de la Vie Artificielle, la notion d'organisation est souvent absente, ou implicitement présente dans la notion d'auto-organisation : « le système s'organise lui-même, il n'est donc pas nécessaire que nous nous en occupions. » En effet, on ne peut pas manipuler les concepts de rôle ou de fonctionnalité au niveau des agents d'un système réactif, car il s'agit de descriptions symboliques, souvent crées par l'observateur ou par le concepteur. Si on veut appliquer une telle notion d'organisation à des SMA réactifs, on est obligé d'effectuer - au niveau du concepteur - une traduction entre le niveau des rôles et le niveau des agents, ce qui limite la dynamique organisationnelle. Une telle approche, qui peut être utile dans des cas spécifiques, est illustrée dans les travaux de Christof Baeijs [Baeijs 98].

Par contre, si on veut expliciter, dans un SMA réactif, le concept d'organisation, et ceci au niveau des agents plutôt qu'au niveau de l'observateur, on est obligé d'utiliser une notion d'organisation liée au positionnement relatif des agents dans l'espace qu'ils occupent. Par exemple, dans le cas des *elastic patterns* [Van Aeken 90a] [Van Aeken 90b] [Baeijs 96], on peut définir l'organisation des agents comme leur positionnement dans l'espace qui constitue leur environnement. Puis, l'évolution du système vers une certaine forme correspond à l'auto-organisation du système. Notons que, dans cet exemple, il s'agit d'un espace en deux dimensions ressemblant beaucoup à l'espace physique. En général, l'espace des agents est d'une nature beaucoup plus virtuelle et complexe.

Spécifiquement, dans le cas des Systèmes Multi-Agents Minimaux, l'espace des agents correspond à leur structure, que nous avons déclaré, non par hasard, synonyme à leur organisation. L'intérêt de notre modèle est dans le fait que la structure des agents représente le **groupement** des agents, un élément fondamental partagé par toutes les notions différentes d'organisation. Les relations de pouvoir, les rôles, les modules fonctionnels, les positions des agents impliquent tous un groupement spécifique d'agents. Dans ce sens, notre modèle d'organisation est minimal et peut être appliqué à des notions d'organisation réactives ou cognitives.

Dans le cas des systèmes réactifs, notre modèle peut être appliqué en reliant l'espace des agents aux groupements exprimables par des SMAM. L'exemple des particules repoussantes, présenté dans le chapitre II.4, illustre cette approche.

En ce qui concerne les systèmes cognitifs, illustrons comment nous pouvons exprimer, en termes de SMAM, les groupements d'agents impliqués par des organisations définies au niveau des rôles et des liens de pouvoir. Les tableaux 1 et 2 montrent la correspondance entre un nombre d'organisations « classiques » et des SMAM représentant l'organisation des agents au niveau fondamental du groupement. La liste d'organisations

« classiques » est inspirée par [Baeijs 98]. Les agents « personnes », par exemple les individus dans une entreprise, sont représentés par des agents atomiques.

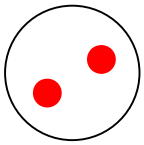
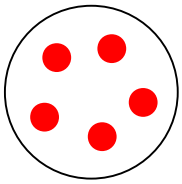
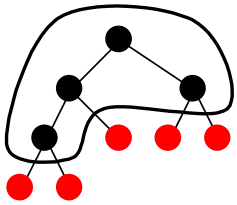
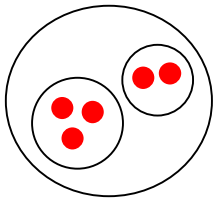
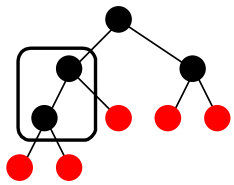
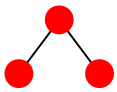
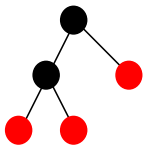
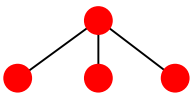
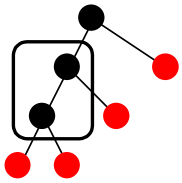
Organisation “classique” — nom	**Organisation “classique” — représentation visuelle**	**SMAM correspondant— représentation arborescente**
membre unique		
couple		
groupe		
groupe multi-niveaux		
hiérarchie simple		
hiérarchie simple — non binaire		

Tableau 1a : Correspondance entre organisations et SMAM (partie 1).

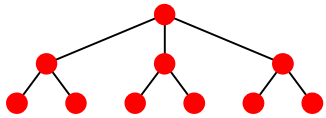
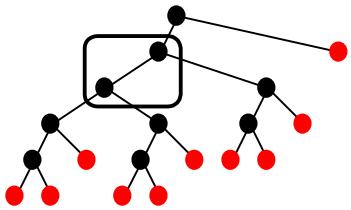
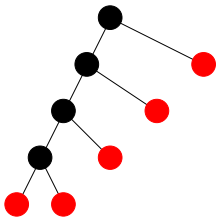
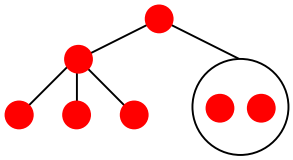
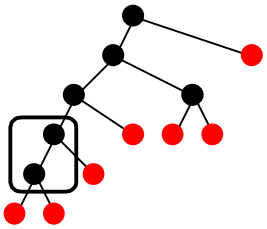
Organisation “classique” — nom	Organisation “classique” — représentation visuelle	SMAM correspondant— représentation arborescente
hiérarchie multi-niveaux		
hiérarchie extrême		
organisation mixte		

Tableau 1b : Correspondance entre organisations et SMAM (partie 2).

Notons que les SMAM ne modélisent que le groupement d’agents et non, par exemple, comment les agents accèdent aux ressources d’information. A chaque organisation « classique » correspond exactement un SMAM. Par contre, à un SMAM donné peuvent correspondre plusieurs organisations « classiques » différentes. La modélisation par SMAM nous permet d’étudier un aspect spécifique d’organisation, mais ne remplace pas - d’une manière évidente - la modélisation « classique ».

Soulignons que la correspondance entre une organisation « classique » et un SMAM n’est pas toujours triviale. Par exemple, à un arbre ne correspond pas forcement un arbre. En effet, la structure binaire des SMAM n’implique pas que le modèle ne puisse modéliser que des organisations hiérarchiques basées sur des arbres (comme celles décrites dans [So 96]).

Les tableaux 1a et 1b peuvent donner l’impression que les organisations hiérarchiques correspondent toujours à des SMAM déséquilibrés et donc instables dans le temps. Ceci est du au fait que, dans ces tableaux, les agents « personnes » sont tous modélisés comme des agents atomiques d’ordre 0. Une modélisation plus réaliste tiendrait compte du fait que les agents placés haut dans la hiérarchie sont, en général, plus complexes que les agents placés bas, et leur modélisation par des SMAM serait plus complexe. Dans ce cas, le SMAM correspondant peut être très équilibré.

Prenons l’exemple de la hiérarchie extrême du tableau 1b. Il s’agit de l’organisation la plus hiérarchique possible : il n’y a que des « chefs » sauf pour un seul « ouvrier » à la fin de la chaîne. Le SMAM correspondant est très mal équilibré : il est, en effet, linéaire. Notons que les deux agents le plus bas dans la hiérarchie se

trouvent au même niveau dans le SMAM : ils forment le groupe des deux moins fortunés. En pratique, un tel système peut être beaucoup plus équilibré que suggéré par le SMAM linéaire, parce que les agents vers le sommet de la hiérarchie sont d'une plus grande complexité que les agents vers le bas.

Donc, afin de modéliser la hiérarchie plus fidèlement, nous devons représenter les agents « personnes » plus complexes par des SMAM également plus complexes. La figure 2 montre la hiérarchie extrême de cinq personnes, mais en modélisant les agents d'une manière plus fidèle. L'agent le plus haut dans la hiérarchie est étiqueté *A5* et modélisé par un SMAM de taille réelle 5. L'agent le plus bas est étiqueté *A1* et compte un seul agent atomique. Notons que les deux agents les plus bas se trouvent toujours au même niveau, mais qu'on peut les différencier sur la base de leur composition différente.

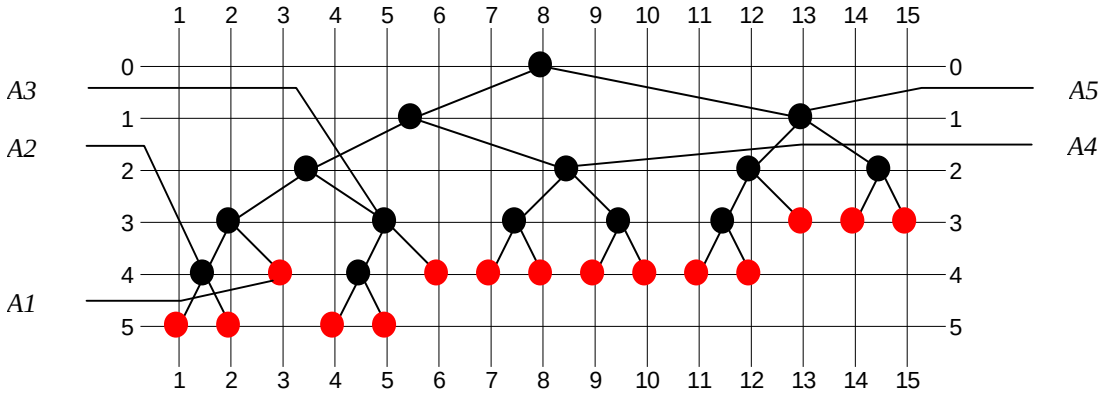


Figure 2 : SMAM correspondant à une hiérarchie extrême.

Cet exemple illustre le fait, et la nécessité, que notre modèle puisse s'appliquer à plusieurs niveaux, d'une échelle « agent » à une échelle « société ». Comment modéliser un agent « personne » par une société d'agents constitue une problématique intéressante et nous espérons de l'étudier plus en détail dans des travaux futurs. Evidemment, ce concept récursif n'est pas nouveau : Marvin Minsky l'a exploré déjà dans les années quatre-vingt [Minsky 86].

Donc, en tenant compte non seulement des liens de pouvoir, mais également de la complexité des agents, les SMAM peuvent être utilisés pour modéliser des aspects importants des organisations « classiques ». Le positionnement et le groupement des agents d'un SMA réactif peuvent également être modélisé par un SMAM. Notons, toutefois, que le modèle des SMAM ne détermine non seulement le groupement des agents, mais également leur évolution à long terme : plus spécifiquement, il exige que les SMAM maximisent leur équilibre dans le temps. La vraie modélisation par SMAM est limitée aux systèmes dotés d'une organisation *dynamique*.

Migration

A l'origine de l'évolution des SMAM se trouvent les migrations des agents. En effet, le seul moyen par lequel un agent peut directement influencer son environnement consiste en sa migration. Dans ce sens, le modèle des SMAM constitue un outil excellent pour étudier certains aspects de la migration d'agents en général.

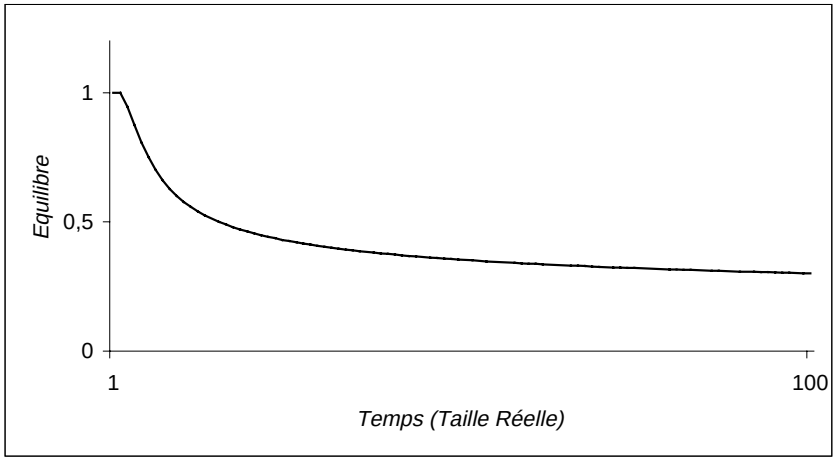
Nous distinguons la migration des agents de la mobilité des agents. La migration d'un agent implique le déplacement de cet agent dans son environnement. La mobilité, par (notre) définition, implique - au niveau *physique* - le déplacement d'un agent d'un site informatique à un autre site informatique. En principe, les notions de migration et de mobilité sont complètement indépendantes. En réalité, elles sont souvent liées parce que - en général - plus le groupement virtuel des agents (dans un SMAM, synonyme à leur organisation) correspond à leur distribution physique, plus le SMA peut fonctionner efficacement. Ceci est une conséquence du fait que - au niveau conceptuel - les interactions intra-groupe sont souvent plus fréquentes que les interactions inter-groupe. Nous étudierons la mobilité plus en détail dans le chapitre III.3.

Tout SMA ne permet pas la migration des agents. Pour que les agents d'un SMA puissent migrer, le SMA doit supporter une **organisation dynamique** et il doit être établi un **protocole de migration**. Nous spécifierons un protocole général de migration dans le chapitre III.3. Le protocole spécifique que nous utilisons dans l'application distribuée *FRIENDS Online MAI* est détaillé dans le chapitre IV.2, dédié à *FRIENDS Online*.

Pour que la migration soit possible, les agents eux-mêmes doivent être dotés de **capacités** de **migration** et de **perception**. La migration aveugle est, en général, peu utile. Si l'environnement dans lequel les agents migrent est de nature sociale (comme l'environnement des SMAM), ils doivent également être dotés de capacités d'**apparence**. En effet, dans ce type d'environnement, la perception est directement liée à la manière dont les agents se montrent aux autres. Par exemple, un agent qui ne se montre pas, ne peut pas être perçu. La capacité d'apparence est loin d'être triviale, et elle implique des questions d'honnêteté et d'abstraction. En effet, même si un agent veut se montrer comme il est, l'exhibition totale de l'agent est souvent trop coûteuse d'un point de vue informatique, ou même impossible si l'agent n'a que des capacités limitées de réflexion. Il s'agit d'un problème important et intéressant mais, cependant, peu étudié dans le domaine. Dans le cas des SMAM, les agents peuvent se manifester en présentant leur entropie, très facile à calculer à partir de l'entropie des fils et constituant une bonne abstraction de l'agent. Evidemment, lorsque plus de détail est nécessaire, une autre méthode doit être utilisée.

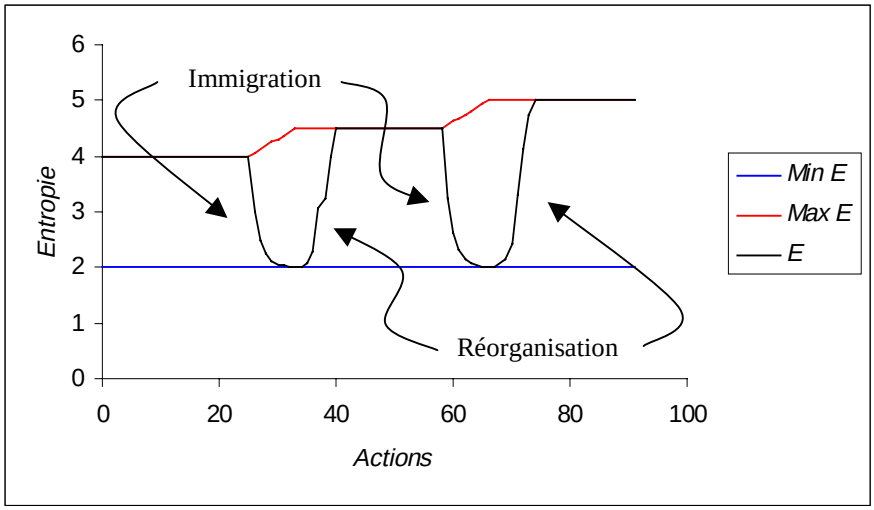
Les SMAM nous permettent d'étudier, à un niveau abstrait, certains phénomènes liés à la migration. Prenons l'exemple d'une entreprise embauchant, d'une manière continue, des nouvelles forces pour répondre à une demande croissante. Si les nouveaux agents ne sont pas intégrés dans l'organisation de l'entreprise, le système deviendra de plus en plus déséquilibré pour - finalement - succomber sous son poids. Essayons de modéliser ce scénario à l'aide des SMAM.

Imaginons qu'un SMAM donné croisse systématiquement en accueillant des agents atomiques immigrants au niveau 0 de la structure. Chaque fois qu'un agent immigre, il forme un couple avec le SMAM existant. Le SMAM résultant est un SMAM linéaire dont la taille croît d'une manière linéaire dans le temps. Par contre, comme le montre le graphique 1, l'équilibre de ce système ne cesse pas de diminuer.



Graphique 1 : Equilibre diminuant d'un SMAM linéaire croissant.

Puisque les différences entre les deux partenaires de chaque couple d'un SMAM linéaire peuvent être très importantes, ces couples peuvent se décomposer facilement. Dans un SMAM ouvert, ceci implique que les agents émigrent facilement et que le système se décompose. Afin d'éviter cette évolution, il est nécessaire que le système se réorganise après chaque instance (ou série) d'immigration. Ceci est illustré par le graphique 2 montrant l'évolution d'un système caractérisé, à la fois, par l'immigration et par la réorganisation. Chaque immigration augmente l'entropie potentielle du système à long terme, mais diminue l'entropie à court terme.



Graphique 2 : Immigration et réorganisation.

De temps en temps, il peut être dans l'intérêt d'un système de ne pas se réorganiser. Prenons l'exemple de la figure 3. L'agent *A* est parfaitement équilibré et l'intégration de l'agent *B* dans sa structure le déstabilise sensiblement à court terme (figure 4) et légèrement à long terme (figure 5). Toutefois, la stabilité du système globale augmentera, à court terme et à long terme, sauf si la situation dans laquelle le SMAM global se trouve (nous sommes dans un contexte ouvert) est tellement instable que même le déséquilibre le plus minime peut mener à des émigrations. Dans ce cas, l'intégration d'un immigrant est inutile, parce qu'elle déstabilise le système deux fois (en l'intégrant et en l'expulsant) et elle n'apporte rien à long terme. Evidemment, il s'agit d'une situation très particulière et, en général, il est dans l'intérêt d'un SMAM d'accepter des immigrants.

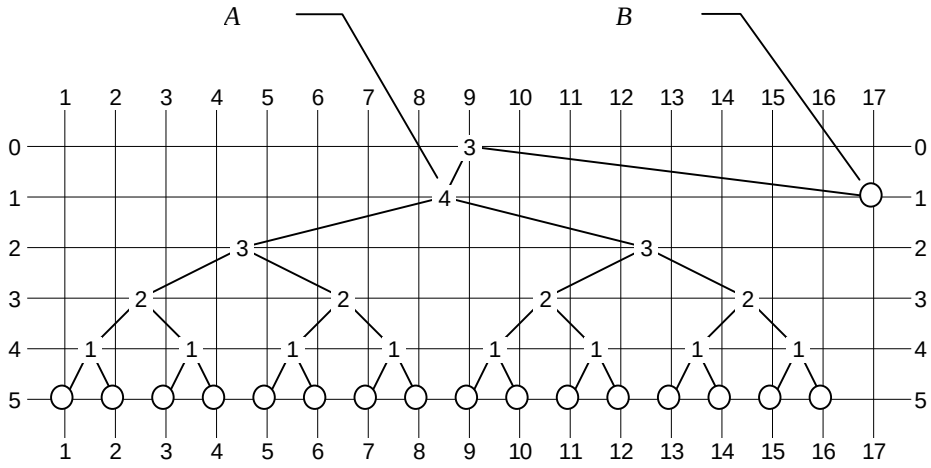


Figure 3 : Intégration de *B* dans *A* (phase initiale).

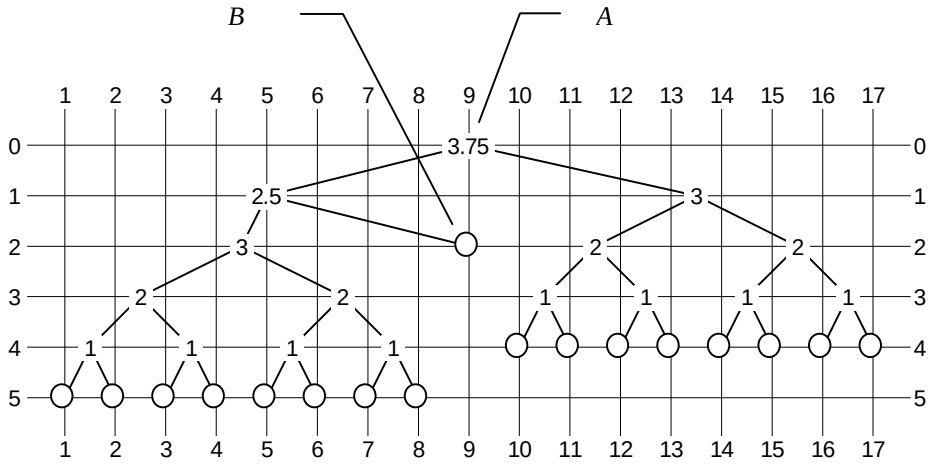


Figure 4 : Intégration de *B* dans *A* (phase intermédiaire).

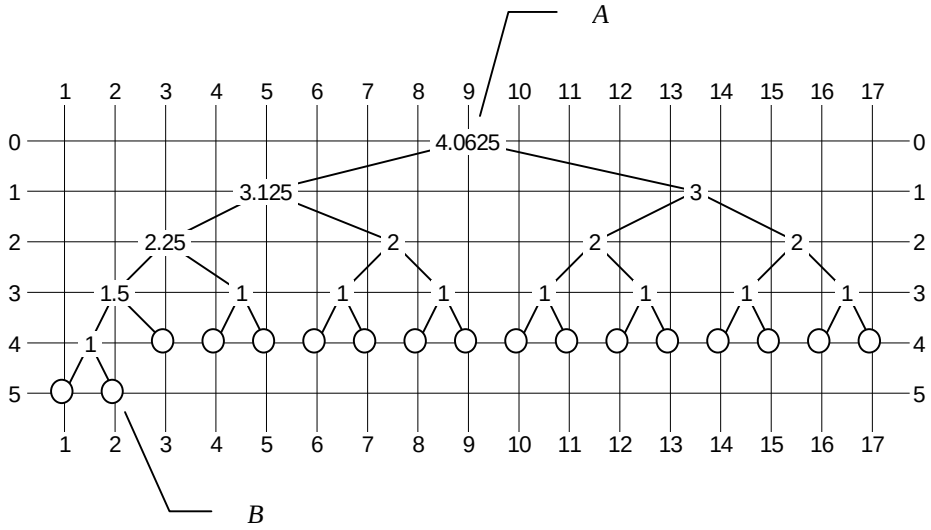


Figure 5 : Intégration de *B* dans *A* (phase finale).

Notons que, dans cette discussion sur la migration il s'agit, par définition, de systèmes ouverts et que notre définition de l'évolution des SMAM fermés n'est pas applicable. Toutefois, nous supposons que les SMAM, fermés ou ouverts, essaient de maximiser leur entropie. Ce comportement peut être formalisé seulement si nous spécifions exactement la nature des SMAM ouverts. Nous espérons étudier cette question dans des travaux ultérieurs.

Donc, malgré sa nature abstraite et minimale, notre modèle peut nous aider à mieux comprendre certains aspects des Systèmes Multi-Agents en général. Le modèle est également lié à la Systémique et à la Vie Artificielle, deux liaisons que nous détaillerons dans le reste de ce chapitre.

II.5.2 Les SMAM et la Systémique

Comme nous l'avons indiqué au début de ce chapitre, au cœur du concept de la Systémique, nous trouvons l'hypothèse qu'il existe des principes universels d'organisation applicables dans tous les domaines scientifiques. Comme le sont la plupart des modèles manipulés dans la Systémique, le nôtre est plutôt abstrait, non lié à une discipline spécifique, sauf au domaine des Systèmes Multi-Agents. Toutefois, puisque nous faisons abstraction de l'implémentation des agents, notre notion des SMA est également abstraite et peut être vue comme un concept systémique. En effet, quel système existant ne peut pas être vu comme un ensemble d'entités actives ?

L'entropie constitue un outil important dans la Systémique. Nous pouvons retrouver sa notion dans un nombre de domaines scientifiques très variés et il est donc peu surprenant de la retrouver dans la Systémique. Situons notre modèle par rapport à cette discipline en illustrant la connexion entre notre notion d'entropie et celle utilisée dans d'autres domaines.

Chris Hillman identifie un nombre de domaines dans lesquels la notion d'entropie est utilisée : la théorie de l'information et du codage, le domaine des systèmes dynamiques, la logique et la théorie des algorithmes, l'inférence statistique et la prédiction, les sciences physiques, les sciences économiques, la biologie et les sciences sociales (et les sciences humaines en général) [entropie]. Notons que, en général, les liens entre les versions différentes de l'entropie sont peu évidents et qu'ils sont surtout liés par le formalisme mathématique et certaines intuitions universelles.

Les domaines classiques appliquant la notion de l'entropie sont les sciences d'information et la thermodynamique. Il est possible de trouver, assez facilement, une connexion entre la notion de Shannon et notre modèle. Au niveau conceptuel, nous pouvons également trouver un lien entre les deux lois de la thermodynamique et la théorie des SMAM.

La théorie de l'information et du codage

La première définition mathématiquement complète de l'entropie a été formulée par Claude Shannon en 1948 [Shannon 48]. Ce travail révolutionnaire a donné la naissance à deux pistes scientifiques différentes, mais très liées : la théorie de l'information (sur la transmission de données) et la théorie du codage (sur le développement de codes).

Shannon, souhaitant mesurer l'information transmise par un médium, a défini l'entropie comme une mesure d'incertitude. Il se posait la question suivante :

Supposons que nous ayons un ensemble d'événements dont les probabilités d'arriver sont p_1, p_2,…,p_n. Est-ce que nous pouvons trouver une mesure indiquant notre incertitude relativement à ces événements ? Cette mesure, disons $H(p_1, p_2,…,p_n)$ doit avoir les propriétés suivantes :

— H doit être continue dans chaque p_i.

— Si tous les p_i sont égaux à $1/n$, H doit être une fonction qui croît de manière monotone en fonction de n. En effet, si tous les événements ont la même probabilité de se passer, plus d'événements implique plus d'incertitude.

— Si un choix peut être décomposé en plusieurs choix successifs, la mesure complète doit être la somme pondérée des mesures individuelles.

Shannon démontre qu'il n'existe qu'une seule mesure H ayant ces trois propriétés. Elle est la suivante :

$$H = -K\sum_{i=1}^{n} p_i \log p_i$$

ou K est une constante déterminant l'unité de la mesure.

Puis, Shannon utilise cette mesure, qu'il appelle *entropie*, pour définir quantitativement des notions telle que la capacité de transmission d'une chaîne. Notons que, plus l'entropie d'une source est grande, plus la capacité de la chaîne doit être grande pour transmettre toutes les informations. En général, l'entropie représente le nombre de bits nécessaire pour coder un ensemble de symboles ayant des probabilités données d'occurrence.

Il n'y pas de rapport direct entre la théorie de Shannon et la théorie des SMAM, car la première concerne des événements et la deuxième concerne des systèmes composés de plusieurs agents. Toutefois, nous pouvons l'utiliser d'une manière indirecte pour démontrer certaines propositions concernant le codage des agents atomiques d'un SMAM par des agents extérieurs. A la base de ces propositions se trouve l'hypothèse que les agents extérieurs parlent plus souvent des agents très visibles du SMAM que des agents peu visibles. Comme mesure de visibilité, nous prenons le niveau de l'agent atomique dans le SMAM. En effet, plus un agent se trouve bas dans un SMAM, plus de groupes, i.e. agents composés, doivent être pénétrés pour y accéder. Formulée d'une manière exacte, notre hypothèse est la suivante :

> Lorsqu'un agent extérieur fait référence à un agent atomique A d'un SMAM S, la probabilité qu'il fasse référence à l'agent A est égale à $2^{-N(A)}$. $N(A)$ est calculé relativement à S.

Sachant cette hypothèse, nous pouvons formuler la proposition suivante :

> L'entropie d'un SMAM est égale au nombre de bits nécessaires pour coder, du point de vue d'un agent extérieur, les références aux agents atomiques individuels.

Cette proposition provient directement de la correspondance entre notre définition d'entropie et celle de Shannon. Elle exprime le fait qu'on peut optimiser le codage des références aux agents atomiques en attribuant peu de bits aux codes référant aux agents très visibles et plus de bits aux codes référant aux agents plus obscurs. Une conséquence directe de la proposition est la suivante :

> Plus un SMAM est équilibré, plus le nombre de bits nécessaires pour coder, du point de vue d'un agent extérieur, les références aux agents atomiques individuels est grand.

Sachant l'évolution naturelle des SMAM, il suit que :

> Un SMAM évolue naturellement vers un état dans lequel le nombre de bits nécessaires pour coder, du point de vue d'un agent extérieur, les références aux agents atomiques individuels est maximal.

Donc, la théorie de Shannon peut être utile lorsqu'on étudie la représentation des agents atomiques par des agents extérieurs.

La thermodynamique

La notion d'entropie a été introduite pour la première fois par Rudolph Clausius, chercheur dans le domaine de la thermodynamique. Plus tard, Ludwig Boltzman a formulé une définition statistique de la notion, liant - jusqu'à un certain degré - l'entropie thermodynamique à l'entropie de Shannon.

Au cœur de la thermodynamique, nous trouvons deux lois qui peuvent être formulées d'un grand nombre de manières différentes. La première loi peut être définie comme suit :

> Le changement de la quantité d'énergie contenue dans un système (pendant un intervalle de temps donné) est égal à la quantité d'énergie introduite dans le système en forme de chaleur (pendant cet intervalle) moins la quantité d'énergie sortie du système en forme de travail (pendant cet intervalle).

Ou, exprimée plus généralement :

> L'énergie d'un système isolé est conservée.

Donc, de l'énergie ne peut pas être créée ou détruite, elle peut seulement être transférée ou transformée. La deuxième loi peut être exprimée comme suit :

> La chaleur ne coule jamais, d'une manière spontanée, d'un objet d'une température inférieure à un objet d'une température supérieure.

En utilisant la notion d'entropie, la deuxième loi peut également être définie comme suit :

> L'entropie totale d'un système isolé ne peut pas diminuer dans le temps.

Dans la thermodynamique, l'entropie est une mesure de « randomness » et la deuxième loi peut être formulée comme suit :

> Tous les processus réels sont irréversibles.

Donc, la deuxième loi postule qu'il est impossible qu'un système isolé évolue d'un état général (plus aléatoire) vers un état spécifique (moins aléatoire).

En ce qui concerne les SMAM, les deux lois expriment très bien les intuitions que rien ne peut être créé à partir de rien et qu'une classe importante de systèmes évoluent d'une manière irréversible d'un état particulier vers un état plus général. Dans ce sens, la correspondance entre les deux lois de la thermodynamique et les deux lois de l'évolution des SMAM n'est pas accidentelle. Soulignons cependant que deux domaines traitent des matières complètement différentes et qu'il s'agit d'une correspondance au niveau conceptuel et non au niveau technique, mais que, dans un contexte systémique, c'est le niveau conceptuel qui a plus d'importance.

II.5.3 Les SMAM et la Vie Artificielle

Christopher Langton, pionnier de la Vie Artificielle et éditeur du journal *Artificial Life*, a défini le sujet de cette discipline comme « *life-as-it-could-be* », une extension de « *life-as-we-know-it* » [Langton 88]. Donc, la discipline concerne premièrement la modélisation, la simulation et la réalisation de systèmes considérés vivants selon une définition donnée.

Autopoièse

Plusieurs définitions sont utilisées et il n'existe pas de consensus général. Toutefois, un modèle en particulier, développé par des biologistes théoriques, est considéré par un grand nombre de chercheurs comme, à la fois, élégant et puissant. Il s'agit du modèle des systèmes auto-poiétiques développé par Humberto Maturana et Francisco Varela [Maturana 73]. Le seul inconvénient du modèle, dans certains contextes, est son haut degré d'abstraction : l'appliquer à des systèmes concrets n'est pas toujours facile.

Le modèle de Maturana et Varela est abstrait - le nôtre l'est aussi. De plus, les deux modèles ont comme sujet des systèmes complexes évoluant d'une manière bien spécifique. Sont-ils liés ? Nous avons étudié cette question, et cette étude nous a aidé considérablement à mieux comprendre, à la fois, notre modèle et le modèle de Maturana et Varela. En effet, nous croyons que les deux modèles peuvent être liés et que, en passant par la théorie de l'autopoièse, notre modèle peut jouer un rôle dans la Vie Artificielle. Nous toucherons à cette dernière hypothèse vers la fin de cette section. Dans un premier temps, regardons la théorie de Maturana et Varela de plus près et essayons de voir comment cette théorie peut être liée à la nôtre.

Dans le contexte de ce mémoire, nous ne pouvons présenter que l'essentiel de cette théorie. Un résumé plus détaillé et très clair, rédigé par Randall Whitaker, qui a appliqué la théorie de l'autopoièse aux entreprises, se trouve à [Observer].

Soulignons d'abord que Maturana et Varela sont des biologistes. A la base de leurs recherches se trouve le désir de construire une théorie de la cognition. Toutefois, à leurs yeux, la cognition est un phénomène biologique : « Cognition is a biological phenomenon and can only be understood as such; any epistemological insight into the domain of knowledge requires this understanding. » [Maturana 80]. Vers la fin de cette section, nous présenterons notre hypothèse, liée à celle de Maturana, que les systèmes artificiellement intelligents doivent également être artificiellement vivants.

Fondamental dans la théorie de l'autopoièse est la notion de l'**observateur**. Un phénomène ne peut être étudié que par un observateur. Donc, la vie également ne peut être identifiée que par un observateur vivant lui-même. Selon Maturana, un observateur est un être vivant capable de distinguer des entités différentes de lui. Les mécanismes nécessaires pour distinguer et identifier des entités vivantes sont donc d'une grande importance.

Maturana spécifie la notion de *distinction* comme suit : « the pointing to a unity by performing an operation which defines its boundaries and separates it from a background. » [Maturana 75].

L'ensemble de relations qui définissent la forme d'un système et qui l'identifient tout au cours de son évolution, Maturana et Varela l'appellent son **organisation** : « The relations that define a machine as a unity, and determine the dynamics of interactions and transformations which it may undergo as such a unity, constitute the organization of the machine. » [Maturana 80].

Durant l'évolution d'un système identifiable, son organisation est constante. Toutefois, sa configuration actuelle peut être différente d'un instant à l'autre. L'ensemble des composants concrets et des relations entre eux est appelé la **structure** du système. Donc, la relation entre structure et organisation est la suivante :

« The organization of a machine (or system) does not specify the properties of the components which realize the machine as a concrete system, it only specifies the relations which these must generate to constitute the machine or system as a unity. Therefore, the organization of a machine is independent of the properties of its components which can be any, and a given machine can be realized in many different manners by many different kinds of components. In other words, although a given machine can be realized by many different structures, for it to constitute a concrete entity in a given space its actual components must be defined in that space, and have the properties which allow them to generate the relations which define it. »

Maintenant, nous pouvons définir la notion d'**autopoièse**. En 1975, Maturana décrit les systèmes autopoiétiques comme suit : « ...autopoietic systems operate as homeostatic systems that have their own organization as the critical fundamental variable that they actively maintain constant. » [Maturana 75]. Donc, un système autopoiétique est un système qui maintien son organisation malgré les perturbations d'un monde dynamique. Plus techniquement, une **machine autopoiétique** est définie comme suit [Varela 89] :

> Un système organisé comme un réseau de processus de production de composants qui
>
> — régénèrent continuellement par leur transformation et leurs interactions le réseau qui les a produits, et qui
> — constituent le système en tant qu'unité concrète dans l'espace où il existe, en spécifiant le domaine topologique où il se réalise comme réseau.

Selon Maturana et Varela, une machine autopoiétique réalisée dans l'espace physique est une machine vivante. Parallèlement, une machine autopoiétique réalisée dans un espace artificiel peut être considérée comme artificiellement vivante.

Maintenant que nous avons défini la notion d'autopoièse, voyons comme la théorie de l'autopoièse peut jouer un rôle dans la théorie des Systèmes Multi-Agents Minimaux, et vice versa. Notons, cependant, que la théorie développée par Maturana et Varela est beaucoup plus élaborée que l'essentiel que nous avons présenté ci-dessus. De plus, plus récemment, Varela a introduit un nombre d'extensions à la théorie. Toutefois, dans le cadre de la thèse présentée, les notions de base nous suffisent. Commençons en identifiant les notions d'organisation et de structure dans le contexte des SMAM.

La notion de structure correspond directement à la notion de structure des SMAM. Comme nous l'avons souligné antérieurement, les SMAM sont *identifiés* par leur structure. Clairement, la notion d'organisation selon

Maturana et Varela ne correspond pas à la notion d'organisation des SMAM, car cette dernière est synonyme à la notion de structure dans les SMAM. Toutefois, nous pouvons identifier, dans le monde des SMAM, une notion qui correspond de très près à la notion d'organisation autopoiétique. Cette identification est à la base du lien entre la théorie de Maturana et Varela et la nôtre. Nous avons appelé cette notion la **signature organisationnelle**. Illustrons la à l'aide de quelques exemples.

Les trois SMAM de la figure 6 ont tous la signature organisationnelle de la figure 7. Les relations entre les composants ne changent pas de SMAM à SMAM, bien que les composants mêmes sont différents (en taille). La figure 8 montre trois autres SMAM ayant la signature organisationnelle de la figure 9.

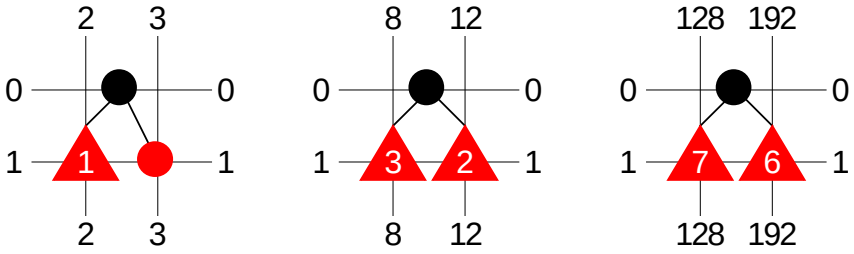


Figure 6 : Trois SMAM ayant la signature organisationnelle de la figure 7.

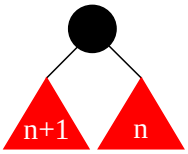


Figure 7 : Exemple d'une signature organisationnelle.

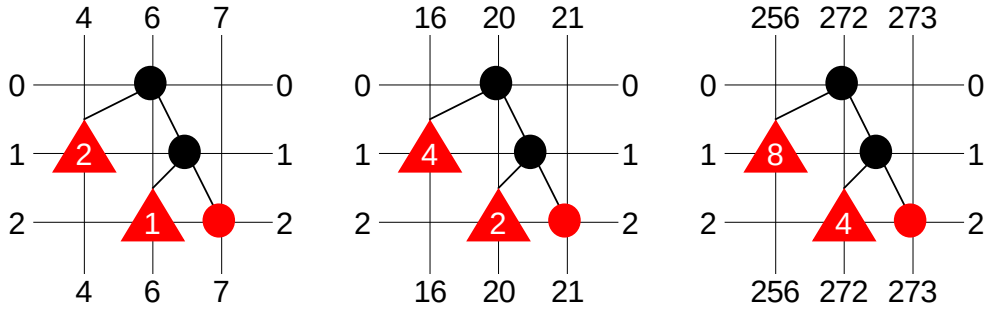


Figure 8 : Trois SMAM ayant la signature organisationnelle de la figure 9.

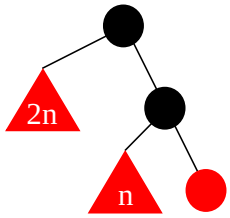


Figure 9 : Deuxième exemple d'une signature organisationnelle.

En général, la notion de signature organisationnelle incorpore un certain degré de flexibilité. Par exemple, on peut considérer les SMAM des figures 10 et 11 comme ayant respectivement la signature organisationnelle des figures 7 et 9. Pour l'instant, nous n'avons pas formalisé cette notion de flexibilité.

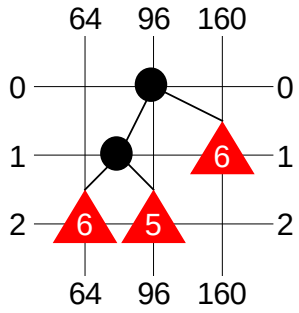


Figure 10 : SMAM ayant une signature organisationnelle similaire à celle de la figure 7.

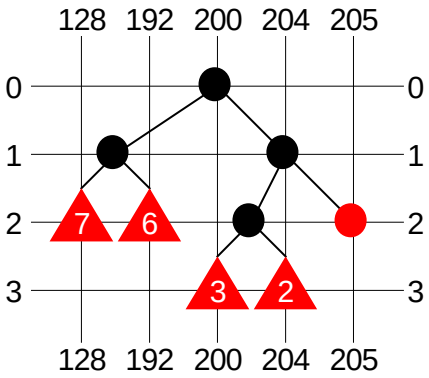


Figure 11 : SMAM ayant une signature organisationnelle similaire à celle de la figure 9.

La question de savoir jusqu'à quel degré une organisation donnée est conforme à une signature organisationnelle donnée est directement liée à la problématique de l'observateur. Si nous voulons appliquer les SMAM à l'étude de l'autopoièse, la première chose à faire est de définir formellement un observateur qui reconnaît certaines « formes » de vie et qui en rejette d'autres. Il s'agit d'un filtre sophistiqué qui ne laisse passer que les SMAM conformes à une signature donnée.

Une fois l'observateur (ou les observateurs) défini, nous devons établir le SMAM initial et l'algorithme d'évolution utilisé. Puis, nous laissons évoluer le système tout en activant l'observateur qui traque les formes de vie pour lesquelles il est programmé.

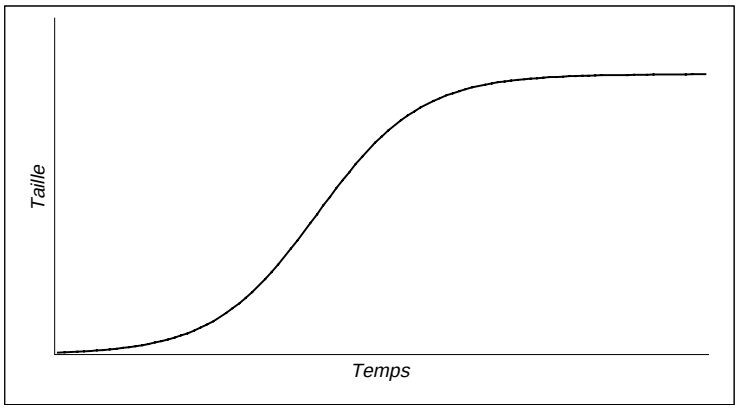
Nos travaux, dans le cadre de notre thèse, s'arrêtent à cette suggestion d'une méthodologie pour l'étude de l'autopoièse à un niveau très abstrait. Nous n'avons effectué l'expérience que pour un cas très spécifique, décrit dans la sous-section suivante. Toutefois, nous sommes convaincus qu'il s'agit d'une piste intéressante.

En effet, contrairement à beaucoup de modèles dans la Vie Artificielle, le nôtre ne produit pas à priori la vie artificielle. Il est bien possible que, si on utilise des algorithmes simples, l'approche nécessite des SMAM de taille et de complexité considérables. De plus, par la définition même des SMAM, toute entité, les agents

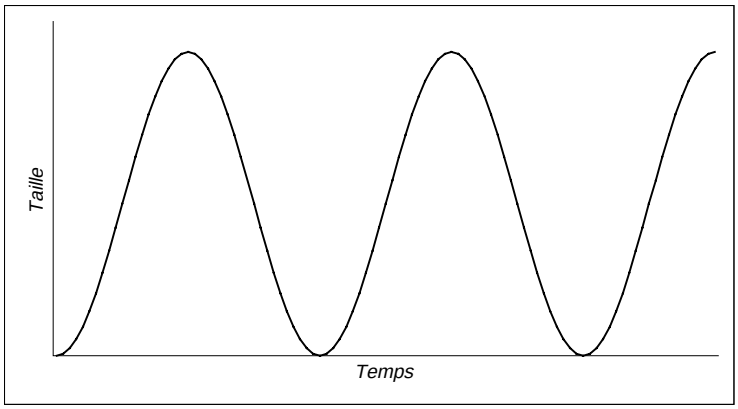
atomiques d'ordre supérieur exclus, se décompose après un nombre d'actions fini. Les entités doivent maintenir leur forme malgré la tendance du système à augmenter son entropie. Dans un sens très précis, ils doivent continûment battre la « thermodynamique artificielle » des SMAM qui mènera inévitablement à la mort de tout système sauf le plus équilibré. Cette correspondance avec la vie physique constitue l'intérêt principal de l'approche.

Maximisation de la taille des SMAM

A un certain moment dans l'évolution historique de notre modèle, nous avions défini l'évolution de la taille des SMAM comme maximisante. En effet, nous n'avions pas défini la notion de SMAM fermé, et nous voulions modéliser la tendance de croître que nous retrouvons dans tellement de systèmes naturels et artificiels. En particulier, les systèmes biologiques et les entreprises suivent, jusqu'à une certaine limite, cette tendance. Ces systèmes croissent jusqu'à un certain seuil, puis ils se réorganisent ou se décomposent, plutôt que diminuer de taille d'une manière systématique. Dans ces systèmes, l'évolution de la taille suivra, par exemple, plutôt une sigmoïde, affiché dans le graphique 3, qu'une sinusoïde, visualisé dans le graphique 4.



Graphique 3 : Sigmoïde.



Graphique 4 : Sinusoïde.

Nous étions obligés d'introduire la notion de SMAM fermé afin d'être capable d'étudier les SMAM d'une manière contrôlée : on ne peut pas dire grand chose d'un système qui est soumis à des influences externes non modélisées. Puis, nous avons défini, en suivant une intuition qui nous semble assez universelle, que la taille d'un

SMAM fermé est constante dans le temps. Cette définition n'est-elle pas une contradiction directe de la définition originale, exprimant elle aussi une intuition importante ?

La solution à cette question se trouve dans l'idée que, au sein d'un SMAM fermé, nous pouvons observer la croissance de la taille de *certaines entités ayant une signature organisationnelle spécifique*. Ce n'est pas le SMAM qui maximise sa taille, mais ces entités observables et autopoiétiques qui émergent du SMAM. En appliquant la théorie de Maturana et Varela, les notions de taille constante et taille croissante peuvent être jointes.

A la base de notre confusion initiale résidait le fait que, dans des SMAM évoluant selon certains algorithmes, nous pouvons très facilement observer des entités qui, en se joignant, maximisent leur taille, à savoir les agents atomiques d'ordre supérieur. De plus, en utilisant ces algorithmes, ces entités maintiennent inconditionnellement leur signature organisationnelle et exhibent une forme très simple d'autopoièse.

Les figures 12, 13 et 14 montrent l'évolution d'un SMAM selon l'algorithme « local aggregated ». Le tableau 2 et le graphique 5 montrent l'évolution de la taille moyenne des agents atomiques d'ordre supérieur. Nous soulignons que les agents atomiques d'ordre supérieur ne se décomposent pas pendant cette évolution.

Notons la similarité entre la forme du graphique 5 et la forme générale du sigmoïde. Ceci semble exprimer la régularité que le gain moyen de taille est - au début - modeste parce que les gains individuels sont faciles à réaliser mais petits, et - vers la fin - également modestes parce que les gains sont importants mais durs à réaliser. Entre ces deux extrêmes, le gain moyen atteint un maximum. Nous n'avons pas étudié ce phénomène de près. Des études intéressantes de ce phénomène restent donc à réaliser.

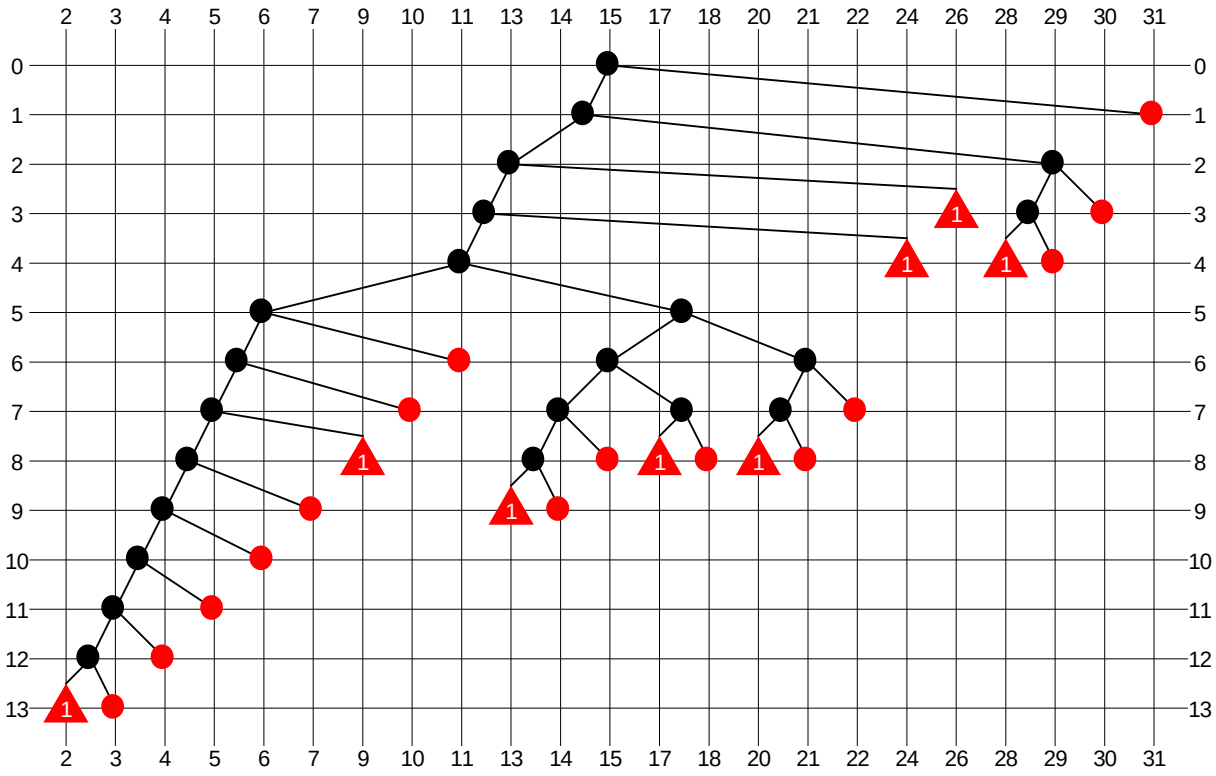


Figure 12 : Evolution d'un SMAM (phase initiale).

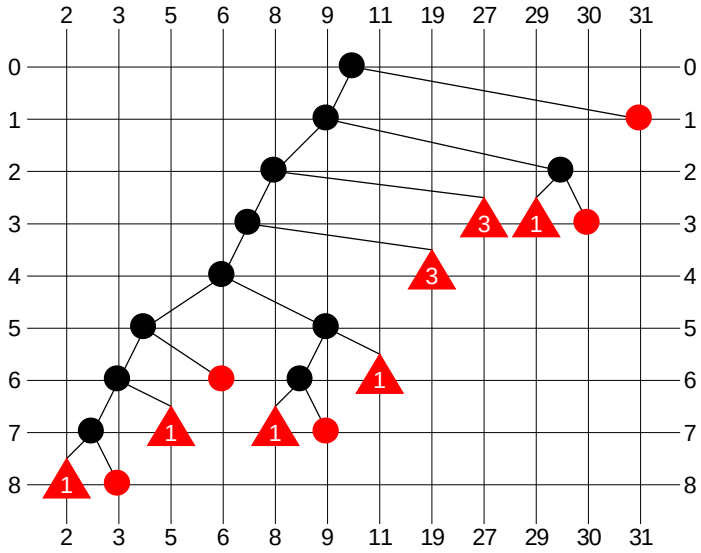


Figure 13 : Evolution d'un SMAM (phase intermédiaire).

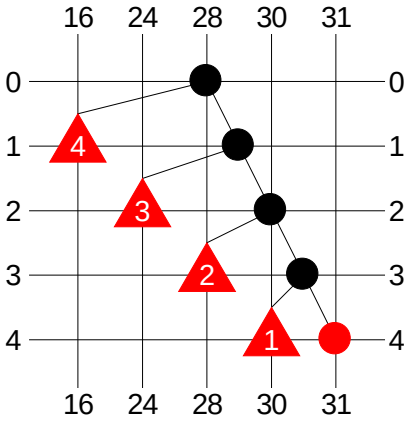
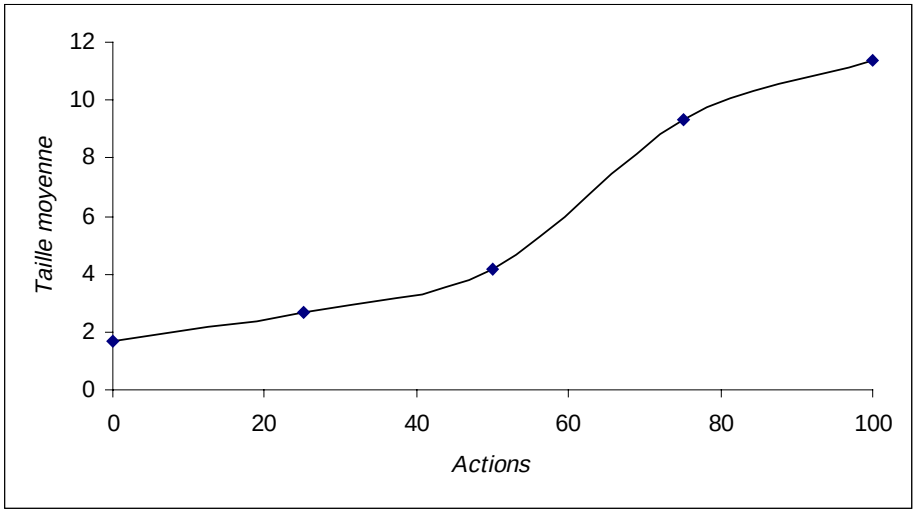


Figure 14 : Evolution d'un SMAM (phase finale).

	ordre	4	3	2	1	0	
	taille réelle	16	8	4	2	1	
	taille	31	15	7	3	1	
temps							***taille moyenne***
0		0	0	0	8	15	1,695652174
25		0	0	3	5	9	2,647058824
50		0	2	0	5	5	4,166666667
75		1	0	3	1	1	9,333333333
100		1	1	1	1	1	11,4

Tableau 2 : Croissance de la taille moyenne des agents atomiques d'ordre supérieur.



Graphique 5 : Croissance de la taille moyenne des agents atomiques d'ordre supérieur.

La Thermodynamique Artificielle

Les domaines de la Vie Artificielle et de l'Intelligence Artificielle ont beaucoup de choses en commun. Tous les deux ont comme but principal la construction de systèmes artificiels ayant certaines propriétés associées typiquement à des systèmes naturels. Tous les deux partagent également le but de mieux comprendre certains systèmes naturels, et ceci en construisant des systèmes artificiels. Toutefois, pour certains auteurs, la ressemblance est essentiellement superficielle et les deux domaines ne sont pas intrinsèquement liés [Heudin 94]. Nous ne partageons pas l'avis de ces auteurs. Pour comprendre nos motivations, il suffit de se poser la question suivante, surprenante mais importante : « **Un système mort peut-il être intelligent ?** ».

Lorsqu'on pose cette question à des scientifiques, ils demandent souvent une définition de la notion de mort. Une définition possible est la suivante : les systèmes morts sont les systèmes qui ne sont pas vivants. Dans ce cas, nous avons défini deux ensembles dont l'intersection est vide et dont l'union contient tous les systèmes possibles. Dans ce cas également, nous pouvons utiliser une définition du vivant, comme celle de Maturana et Varela, pour définir la mort. Mais, est-ce qu'un système non vivant est toujours mort ? Dans un contexte folklorique, nous sommes tous familiers avec le concept de « zombie », synonyme de « undead ». Ces entités ne sont ni mortes ni vivantes : elles suivent un comportement bien défini, un ensemble de règles, mais il leur manque une âme. Il s'agit des caricatures du vivant.

Dans plus d'un sens, les systèmes d'IA classiques ressemblent beaucoup à ces entités « zombie ». Ils ne sont pas morts, parce qu'ils bougent et agissent. Ils ne sont pas vivants, parce qu'ils n'ont pas d'âme. Il nous semble que c'est exactement cette caractéristique qui a motivé des gens comme John Searle à critiquer l'Intelligence Artificielle. Prenons l'exemple de la pièce chinoise formulé par Searle [Searle 80]. Dans cette pièce se trouve une personne, ne comprenant pas le chinois, qui reçoit une série de symboles chinois de l'extérieur et les transforme, en suivant un nombre de règles strictes, en une autre série de symboles chinois. Puis, la personne envoie la nouvelle série à l'extérieur, où elle est interprétée comme une réponse à la série originale

Selon Searle, cet exemple illustre pourquoi un PSS (Physical Symbol System [Newell 80]), même s'il passe le test de Turing, ne peut jamais être intelligent : la personne manipulant les symboles ne comprend pas ce qu'il fait, il n'y a pas de vraie compréhension.

Nous croyons que Searle essaie de communiquer une intuition correcte, mais qu'il le fait d'une manière erronée. L'erreur principale consiste à choisir un manipulateur au sein de la pièce. Exactement la même erreur a été commise il y a quelques siècles dans le contexte de l'étude de la vision. Les chercheurs venaient de découvrir que l'image du monde était projetée inversement sur la rétine et se posaient la question pourquoi nous ne percevons pas le monde à l'envers. A la base de leur confusion résidait le fait qu'ils plaçaient, implicitement ou explicitement, un petit homme dans la tête observant les données venant des capteurs. Ils imaginaient un homunculus dans l'homme.

Plus tard, les chercheurs ont compris que la notion était erronée et que la perception et la conscience doivent être vues comme des phénomènes émergents. Toutefois, dans d'autres contextes, comme celui de l'Intelligence Artificielle, la notion d'homunculus revient de temps en temps.

Searle exprime cependant correctement le sentiment qu'un système sans âme ne peut pas être intelligent, qu'une pièce ne peut pas être intelligente. En effet, elle ne peut pas être intelligente parce qu'elle n'interagit pas, d'une manière riche et interactive, avec le monde autour d'elle auquel réfèrent les symboles chinois. Pour que le système soit vraiment intelligent, il est primordial que ces interactions aient lieu jusqu'au niveaux les plus profonds, les plus physiques. A ces niveaux là, ce qui compte n'est pas la traduction de textes, mais la *survie*. Nous croyons que c'est la vie qui donne l'âme aux systèmes non morts, que c'est la vie qui manque dans les systèmes IA actuels.

Dans un contexte d'Intelligence Artificielle, la vie elle aussi peut être artificielle. Ceci implique toujours qu'un système artificiellement intelligent doit continuellement maintenir la cohérence de sa structure dans son monde. Notons que le système et le monde peuvent être virtuels. Stevan Harnad argumente que nous ne pouvons pas nous brûler avec du feu simulé [Harnad 95]. Ceci est vrai. Toutefois, des entités virtuelles peuvent très bien se brûler avec du feu virtuel.

Donc, nous croyons que l'intelligence implique la vie. Toutefois, lorsque nous regardons des systèmes développés par des chercheurs dans la Vie Artificielle, nous avons souvent un sentiment similaire à l'intuition qui nous est communiquée par Searle. Ces systèmes sont fort intéressants, mais souvent ne semblent pas vraiment être vivants. Souvent, on a l'impression qu'il s'agit d'un jeu, plutôt que de survie, et qu'on a créé une caricature de la vie.

Nous croyons que la situation est similaire à celle de l'Intelligence Artificielle et que nous devons aller encore plus loin, jusqu'aux interactions au niveau de la thermodynamique. C'est la thermodynamique qui rend la vie naturelle non évidente et qui fait la différence entre jeu et survie. Prenons, par exemple, la complexité étonnante des être vivants les plus simples : cette complexité minimale est, entre autre, une conséquence directe des contraintes thermodynamiques.

Nous avons indiqué que les SMAM ne modélisent pas directement les entités vivantes selon Maturana et Varela, mais plutôt leurs constituants. Donc, les règles de comportement des SMAM ne sont pas des règles d'un vivant artificiel, mais plutôt des règles d'une thermodynamique artificielle. De ce point de vue, la correspondance entre

les deux lois des Systèmes Multi-Agents Minimaux et les deux lois de la thermodynamique nous convient très bien.

Notre modèle ne contient pas la notion d'énergie et on peut se demander s'il est conseillé de parler d'une instance de la Thermodynamique Artificielle. Il s'agit d'un modèle tellement abstrait que le fait que nous ayons seulement un type principal de « matière » (par exemple, masse) et non deux (par exemple, masse et énergie) ne nous semble pas gênant. Des notions additionnelles peuvent toujours être introduites. De plus, nous n'avons pas développé notre modèle comme étant un modèle du monde physique. Notons cependant que, si jamais nous voulons tenter cette aventure, nous pouvons certainement exploiter l'équivalence de masse et énergie dans le monde physique. Toutefois, dans un premier temps, nous nous limitons au monde informatique.

CHAPITRE III.1

APPLIQUER LES SYSTEMES MULTI-AGENTS MINIMAUX

I hear and I forget, I see and I remember, I do and I understand.
Confucius

III.1.1 La validation de modèles scientifiques

La science est caractérisée par une dimension socioculturelle forte : la partie principale du développement et de la distribution des modèles est effectuée au sein d'une communauté scientifique dont l'interaction avec le monde extérieur est limitée. Par conséquence, la validation initiale des modèles est effectuée par les scientifiques selon des critères socioculturelles (mémétiques, si on veut [Dawkins 76]). Par exemple, un modèle qui élabore un modèle populaire est typiquement plus facilement accepté, qu'un modèle nouveau, très différent des modèles existants. Toutefois, pour qu'un modèle puisse survivre à long terme et être accepté à une échelle plus large, il doit être validé selon des critères relativement indépendants de la dimension socioculturelle de la communauté scientifique. Nous croyons que deux méthodes de validation externe sont directement identifiables : la validation par **prédiction de phénomènes** et la validation par **construction de systèmes utiles**. Un exemple de la première est la prédiction d'une éclipse solaire. La construction d'un laser constitue un exemple de la deuxième.

Evidemment, ces deux méthodes ne sont pas complètement indépendantes. Par exemple, un système utile doit forcement être prévisible afin de garantir sa fonctionnalité voulue. La différence principale entre les deux est que la première concerne des systèmes existants (souvent naturels) et que la deuxième concerne des systèmes à construire (toujours artificiels). Notons également que ces méthodes ne permettent pas une validation absolument objective. En effet, la première méthode nécessite toujours un nombre de choix, par exemple liés aux mesures, étant plus ou moins arbitraires. En ce qui concerne la deuxième méthode, la même notion d'utilité n'est pas nécessairement partagée par tout le monde. Toutefois, malgré ces défauts, une validation par une de ces méthodes nous semble très utile.

III.1.2 Les SMAM augmentés

Notre thèse se situe dans le contexte de l'informatique et elle implique automatiquement des systèmes artificiels. Nous avons essayé de valider notre modèle par la construction de systèmes utiles.

Les SMAM sont des systèmes abstraits et leur application, sans aucune modification, à des problèmes concrets n'est pas évidente. Une des seules applications directement identifiables consiste en la transformation de la représentation d'un nombre de base 1 à base 2, et ceci - dans le cas des SMAM purs - pour une classe très restreinte de nombres. La figure 1 montre, en utilisant des SMAM, la représentation en base 1 et la représentation en base 2 du nombre 7. Notons que le nombre représenté par un SMAM est égal à la taille réelle de ce SMAM. La représentation en base 1 correspond à un SMAM linéaire. La représentation en base 2

correspond à un SMAM composé d'un nombre d'agents atomiques d'ordre supérieur, tous différents et tous représentant une puissance de deux.

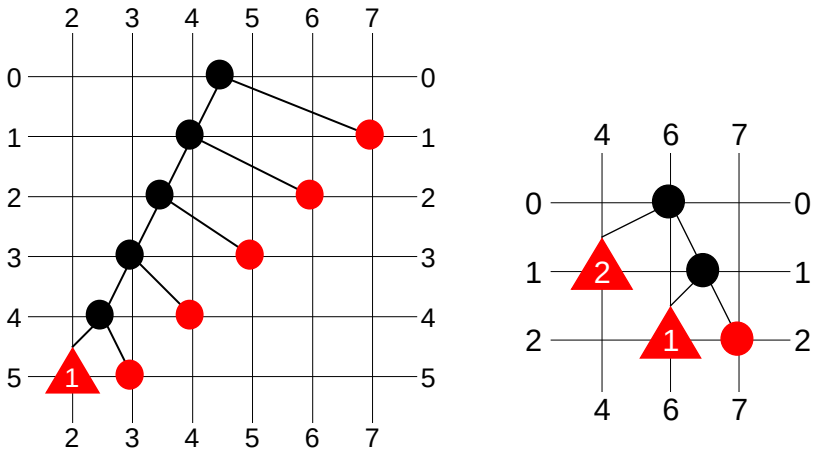


Figure 1 : Représentation en base 1 et en base 2 du nombre 7.

L'algorithme « global aggregated », décrit dans le chapitre II.2, effectue exactement la transformation de base 1 à base 2. Mais, comme nous l'avons vu, cet algorithme n'implémente pas le comportement propre des SMAM. En effet, seuls les SMAM représentant des puissances de 2 évoluent naturellement d'une représentation en base 1 à une représentation en base 2. Ceci n'est pas nécessairement le cas pour les autres nombres. Par exemple, n'importe quel algorithme implémentant correctement l'évolution naturelle des SMAM, transforme tout SMAM représentant le nombre 22 en le SMAM de la figure 2, qui ne correspond pas à la représentation en base 2. L'algorithme « global aggregated », par contre, produit le résultat désiré de la figure 3.

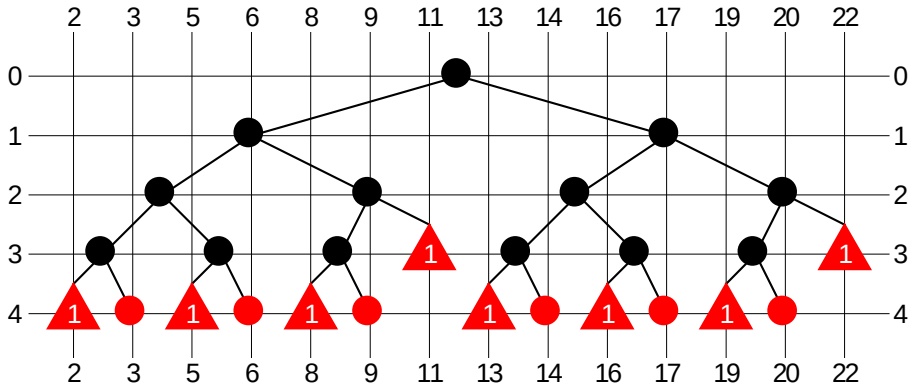


Figure 2 : SMAM de taille réelle 22 et d'équilibre maximale.

Lorsqu'un SMAM évolue selon l'algorithme « global aggregated », seuls les agents atomiques d'ordre supérieur peuvent migrer. De plus, jamais les agents atomiques d'ordre supérieur ne se décomposent. Donc, en restreignant le comportement des agents, l'évolution naturelle du SMAM est limitée, mais une fonctionnalité spécifique est obtenue. Ceci est le principe de base que nous appliquerons pour rendre les SMAM utiles. C'est en limitant leur liberté que nous pouvons les rendre utiles.

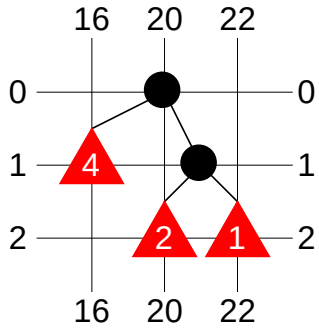


Figure 3 : Représentation binaire du nombre 22.

Les agents atomiques d'ordre supérieur de l'exemple ci-dessus sont, dans plus d'un sens, des **symboles** représentant des concepts d'un monde extérieur. En général, nous pouvons définir un nombre arbitraire de SMAM spécifiques représentant des symboles externes. Le tableau 1 montre un exemple d'une « translation » spécifique.

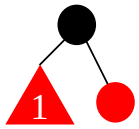
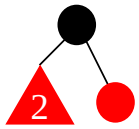
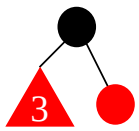
Symbole	SMAM
A	
B	
C	

Tableau 1 : SMAM représentant des symboles externes.

Puis, nous pouvons construire des SMAM à partir de ces symboles et restreindre leur évolution d'une telle manière que les symboles ne se décomposent jamais. La figure 4 montre un SMAM composé de deux 'A', deux 'B' et quatre 'C'. L'évolution restreinte du SMAM le mène au SMAM de la figure 5. Dans le SMAM final, tous les symboles identiques sont groupés. En effet, les agents suivent toujours le principe « qui se ressemble, s'assemble », sauf s'ils font partie d'un symbole, une condition qui les rend inactifs.

En associant des symboles à des SMAM spécifiques, nous pouvons construire toute une classe de systèmes utiles. Comme nous le verrons plus tard, ces systèmes effectuent une forme de clustering qui peut être appliqué dans des contextes différents. De plus, ces systèmes sont naturellement distribués, puisque - au niveau conceptuel - les agents suivent individuellement et de manière autonome le comportement « qui se ressemble, s'assemble ».

Les SMAM dans lesquels nous avons introduit - en restreignant leur comportement - des symboles, nous les appelons des **SMAM augmentés**. Différents des SMAM purs, les SMAM augmentés ne maximisent pas nécessairement leur entropie, mais réalisent des fonctionnalités voulues. En général, ils maximisent une forme d'équilibre non seulement dépendante de la forme absolue des SMAM, mais également de la distribution des symboles dans les systèmes.

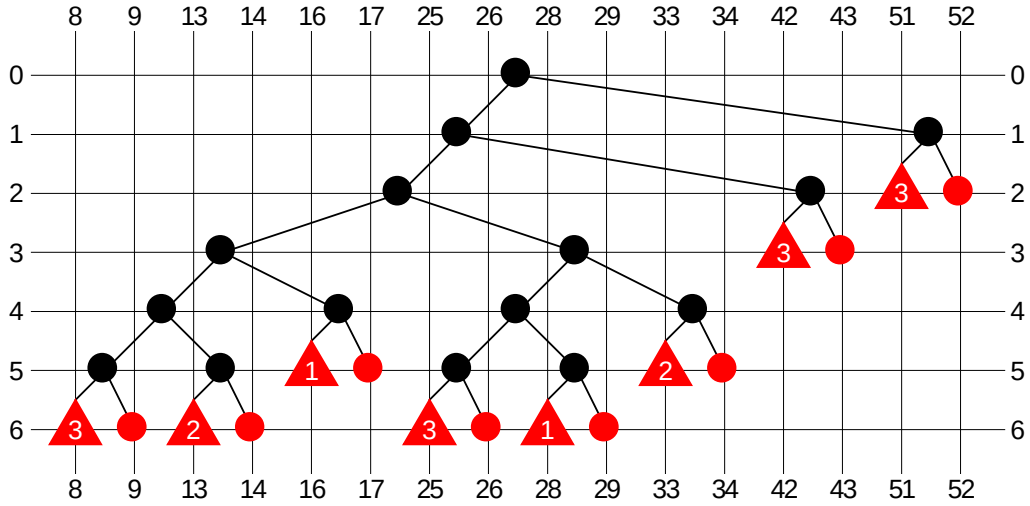


Figure 4 : SMAM restreint, représentant plusieurs symboles.

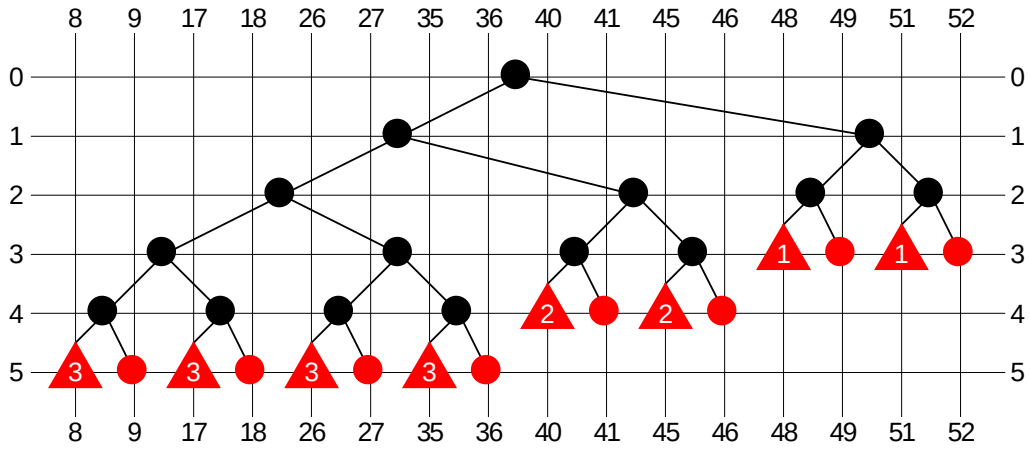


Figure 5 : Groupement de symboles dans un SMAM restreint.

Les symboles circulant dans les SMAM augmentés ne doivent pas forcement être représentés par des SMAM fixes. Ils peuvent être ajoutés, comme des *attributs*, directement aux agents atomiques et composés. Dans ce cas, chaque agent et doté d'un attribut qui représente un ou plusieurs symboles. Le fait que les agents composés sont également équipés d'un attribut les aide à trouver des agents similaires. En effet, l'attribut d'un agent composé consiste typiquement en l'ensemble de symboles partagés par tous les agents faisant partie de l'agent. Nous illustrerons cette approche dans la section suivante où nous introduirons « FRIENDS », notre application de base des SMAM.

Concluons cette section en soulignant que les agents d'un SMAM n'interprètent pas les symboles qui leur sont attribués. Ces symboles font partie de leur identité et sont utilisés dans l'évaluation de la ressemblance entre

deux agents. Donc, le problème de l'enracinement des symboles ne se pose pas, car ils sont comparés, plutôt qu'interprétés.

III.1.3 FRIENDS

Les applications que nous avons développées sont toutes inspirées du problème suivant :

- On nous demande de construire un système organisant automatiquement une population de personnes selon les intérêts individuels. Dans l'organisation, les personnes ayant des intérêts en commun doivent se trouver proches l'une de l'autre.
- De plus, nous devons nous attendre à une dynamique importante du système. De manière continue, des nouvelles personnes entrent dans le système, et les personnes déjà présentes changent régulièrement leur intérêts.
- Finalement, le système doit pouvoir accommoder plusieurs millions de personnes et doit donc nécessairement être distribué.

Comme nous le verrons, il s'agit d'un problème actuel et difficile. De plus, comme nous le montrons dans cette section, le problème est lié de près au modèle des SMAM. Nous avons développé trois systèmes qui offrent tous, à des degrés différents et de manières différents, une solution au problème formulé ci-dessus. Nous les avons appelé « FRIENDS Offline », « FRIENDS Numbercruncher » et « FRIENDS Online ». Le nom commun « FRIENDS » est inspiré par l'intuition que des amis partagent, jusqu'à un certain degré, les mêmes intérêts.

FRIENDS Offline est une plate-forme expérimentale que nous avons utilisée pour expérimenter les SMAM et réaliser la transition de la théorie aux applications « industrielles ». FRIENDS Numbercruncher est conçu pour des populations statiques, mais mène à des solutions de grande qualité pour des populations de taille relativement élevée. FRIENDS Online offre une solution relativement complète au problème, et ceci dans le contexte de l'Internet. Tous les trois sont directement basé sur le concept des SMAM. Comment est-il appliqué ?

D'abord, nous avons formalisé les intérêts des individus. Il s'agit d'un problème important, étudié - entre autre - dans le domaine de *Community Ware* [Ishida 97]. Dans FRIENDS, nous formalisons les intérêts d'une personne comme un vecteur de mots-clés. Un mot-clé est une séquence de caractères. Nous nous rendons compte que cette approche n'est pas seulement très populaire, mais également très limitée. Deux problèmes principaux peuvent être indiqués. Primo, des personnes différentes peuvent utiliser des mots-clés différents pour designer les mêmes concepts. Secundo, beaucoup d'intérêts ne peuvent pas être exprimés par des mots-clés. Par exemple, des goûts en musique ou en peinture ne se traduisent pas, ou très difficilement, en mots. Le premier problème peut être résolu en incorporant un boucle de retour entre le système et les utilisateurs. En consultant le système, les utilisateurs peuvent adapter leurs mots-clés à ceux étant « en vogue ». Nous détaillerons ce principe dans le chapitre IV.2 dédié à FRIENDS Online. Le deuxième problème est plus dur, mais non impossible, à résoudre. On effet, on peut imaginer une interface utilisateur multimédia à l'aide de laquelle l'utilisateur peut indiquer des préférences d'une manière non verbale. Il s'agit d'une problématique fascinante que nous espérons explorer dans le futur.

Ayant une méthode de formalisation des intérêts, nous nous sommes concentrés sur le problème d'organisation. Nous avons pris la décision - très naturelle dans le cadre des SMA - d'associer un agent à chaque utilisateur. Donc, chaque utilisateur est doté d'un agent « représentant » qui représente ses intérêts dans le système.

Puis, afin d'incorporer le concept d'organisation, nous avons introduit les SMAM en associant à chaque utilisateur un agent *atomique* faisant partie d'un seul SMAM représentant le système complet. Les listes de mots-clés (et des informations additionnelles) sont ajoutées - comme des attributs - aux agents atomiques. Les agents composés représentent des groupes composés de deux ou de plusieurs utilisateurs. Chaque agent composé est également doté d'une liste de mots-clés. Cette liste contient les mots-clés en commun entre les deux sous-agents de l'agent. A cause de la récursivité des SMAM, cette liste est composé de tous les mots-clés partagés par tous les agents faisant partie de l'agent. Notons que les SMAM dans FRIENDS sont des SMAM augmentés, plutôt que des SMAM purs. Toutefois, lorsqu'il n'y a pas de confusion possible, nous utilisons le terme « SMAM » tout court pour les désigner.

Donc, l'organisation des utilisateurs réalisée par FRIENDS est identique à la structure du SMAM au sein du système. Puis, par définition, l'évolution naturelle du système le mènera à une organisation d'un équilibre maximal. En effet, le comportement des agents « qui se ressemble, s'assemble » exprime exactement la fonction que nous voulons que le système réalise.

De plus, comme l'organisation d'un SMAM est naturellement dynamique, à tout instant des agents atomiques peuvent être introduits ou enlevés : le système se réorganisera automatiquement. De manière similaire, les mots-clés d'un agent atomique peuvent être changés : il suffit d'enlever l'agent original et d'introduire l'agent modifié.

Evidemment, le problème n'est pas résolu en associant un SMAM à la population des utilisateurs. Nous devons toujours développer l'algorithme faisant évoluer le système comme un SMAM en tenant compte des attributs. Comme notre recherche des SMAM purs nous l'a appris, il s'agit d'un problème peu trivial, surtout si nous souhaitons que le système soit distribué. Dans une première phase, nous avons utilisé FRIENDS Offline pour expérimenter le concept et des algorithmes différents. Dans une deuxième phase, nous avons construit FRIENDS Numbercruncher et FRIENDS Online, en utilisant des algorithmes centralisés. Dans une troisième phase, nous avons construit FRIENDS Online « Mobile Agent Implementation » pour établir une piste vers la réalisation de versions réellement distribuées. Toutes ces applications sont détaillées dans les chapitres suivants.

CHAPITRE III.2

FRIENDS OFFLINE, OUTIL D'EXPERIMENTATION

Only experiments test theories.
Walter Tichy

III.2.1 Une plate-forme interactive

Les principes de base des Systèmes Multi-Agents Minimaux sont extrêmement simples. Toutefois, ils suffisent pour construire des systèmes extrêmement complexes. Cette situation est similaire à celle de la construction de programmes informatiques ou tout autre système complexe : à partir d'un nombre limité de composants et de règles, des systèmes de grande complexité peuvent être construits.

La construction de tels systèmes est typiquement itérative. Personne ne demande à une équipe de programmeurs de construire, sans tester, un programme de grande taille sans s'attendre à un certain nombre de bogues. La programmation suit un cycle itératif composé - entre autre - du développement et de la validation (ou *testing*). La raison principale derrière cette approche est le fait qu'un système complexe ne peut pas facilement être réduit à un ensemble d'équations ou à toute autre entité symbolique manipulable à un niveau abstrait. De plus, la technologie informatique facilite de plus en plus l'expérimentation interactive de systèmes complexes, ce qui a également augmenté l'intérêt de l'expérimentation par rapport à l'analyse symbolique.

Cette évolution n'est pas toujours positive. Beaucoup de systèmes complexes, comme certains systèmes informatiques sont parfaitement analysables d'une manière symbolique et cette analyse ne doit pas être remplacée par l'expérimentation. Toutefois, une classe importante de systèmes ne se laisse pas analyser à l'aide des outils symboliques connus. Par contre, la puissance croissante des ordinateurs numériques nous permet de les étudier d'une manière expérimentale. Des méthodes expérimentales, jadis réservées aux sciences empiriques telles que la physique ou la chimie, sont maintenant appliquées dans les mathématiques et des sciences nouvelles telles que la Vie Artificielle ou les Systèmes Multi-Agents.

C'est dans ce contexte que se situe notre étude et développement des Systèmes Multi-Agents Minimaux. D'un côté se trouve la formalisation et l'analyse symbolique du modèle, de l'autre côté se trouve l'expérimentation. Les deux sont développés parallèlement et ont produit - à la fois - le modèle formel tel qu'il est présenté dans la première partie de cette thèse, et la plate-forme expérimentale « FRIENDS Offline » que nous présentons dans ce chapitre. A part avoir joué un rôle important dans le développement du modèle, la plate-forme nous a été très utile dans la transition de la théorie vers les applications. En effet, elle nous a permis d'expérimenter le concept de FRIENDS à un niveau abstrait.

Dans le reste de cette section, nous présenterons d'abord l'essentiel de la plate-forme. Puis, nous montrons sa fonctionnalité comme prototype de FRIENDS.

L'essentiel de la plate-forme

Notons qu'un manuel complet de FRIENDS Offline 1.7 se trouve à la fin de cette thèse. Il constitue la première annexe. Dans cette section, nous ne présentons que l'essentiel du programme, version 1.7.

FRIENDS Offline 1.7 est un programme conçu pour Microsoft Windows NT 4.0 et Windows 95/98. Il a été développé à l'aide de Microsoft Visual C++ 5.0. La taille de la version autonome (*Statically Linked*) du code exécutable est 393 Ko.

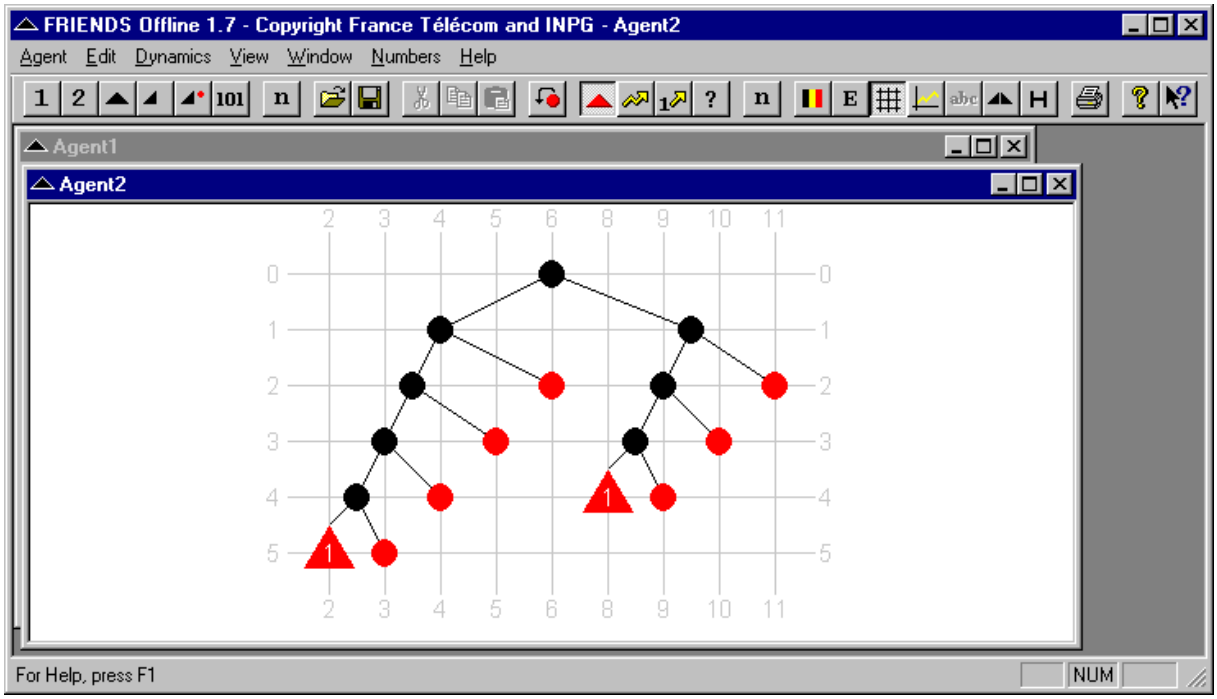


Figure 1 : Fenêtre principale de FRIENDS Offline.

La figure 1 montre une vue typique de FRIENDS Offline. Nous pouvons distinguer la barre de menus composée des éléments « Agent », « Edit », « Dynamics », « View », « Window », « Numbers » et « Help ». Notons que la seule langue actuellement supportée est l'anglais. Toutefois, l'application peut facilement être adaptée pour supporter d'autres langues. Au-dessous de la barre de menus se trouve la barre d'outils composée d'un nombre de boutons affichant des icônes. Les fonctions principales des menus sont dupliquées dans cette barre. Tout en bas de la fenêtre principale se trouve la barre d'état indiquant des informations utiles. Entre la barre d'outils et la barre d'état se trouve la région principale dans laquelle sont affichées les fenêtres associées aux documents. Chaque document représente un SMAM. Notons que plusieurs documents peuvent être ouverts en même temps. En effet, FRIENDS Offline est une application MDI (*Multiple Document Interface*, comme Microsoft Word), plutôt qu'une application SDI (*Single Document Interface*, comme Microsoft Wordpad).

FRIENDS Offline permet aux utilisateurs de **créer** des SMAM (de types différents), de les **éditer**, de les **faire évoluer** et de les **afficher** de manières différentes. De plus, des représentations graphiques des SMAM et de leur évolution peuvent être **exportées** (copier - coller) et **imprimées**.

L'application est préparée pour être équipée des fonctions de sauvegarde et d'aide. Elles ne sont pas implémentées dans la version 1.7, car le nombre limité de chercheurs utilisant cette version n'en a pas eu besoin. Nous envisageons les incorporer dans la version 1.8.

En plus de la fenêtre principale, des fenêtres secondaires peuvent être créées par l'application, typiquement pour afficher des représentations graphiques alternatives ou l'évolution de l'entropie. Les figures 2, 3 et 4 montrent de telles fenêtres. Nous vous renvoyons au manuel de FRIENDS Offline pour une présentation complète de toutes les fenêtre secondaires.

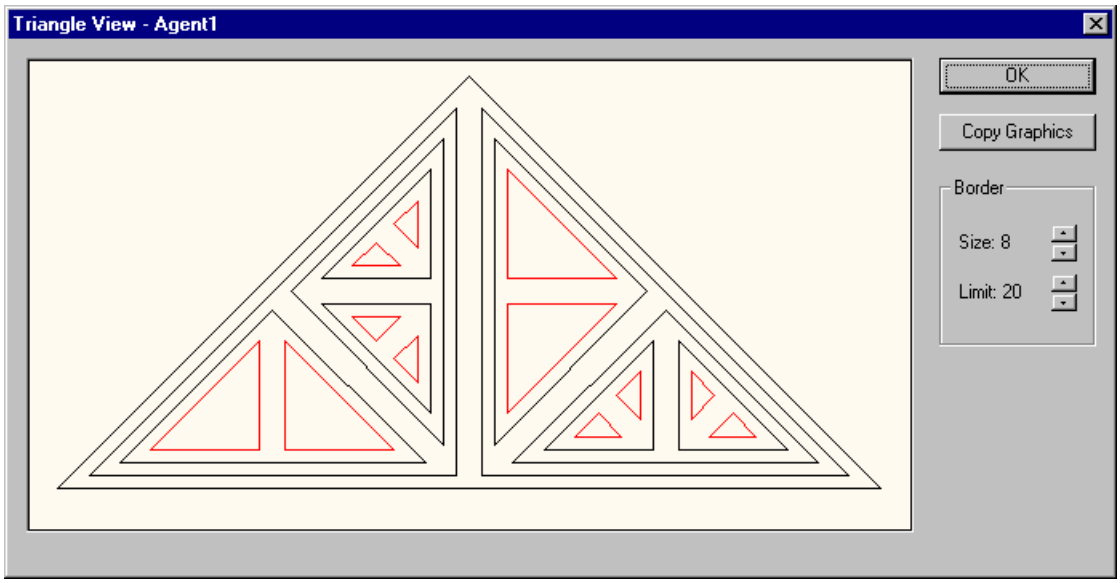


Figure 2 : Fenêtre « Triangle View » de FRIENDS Offline.

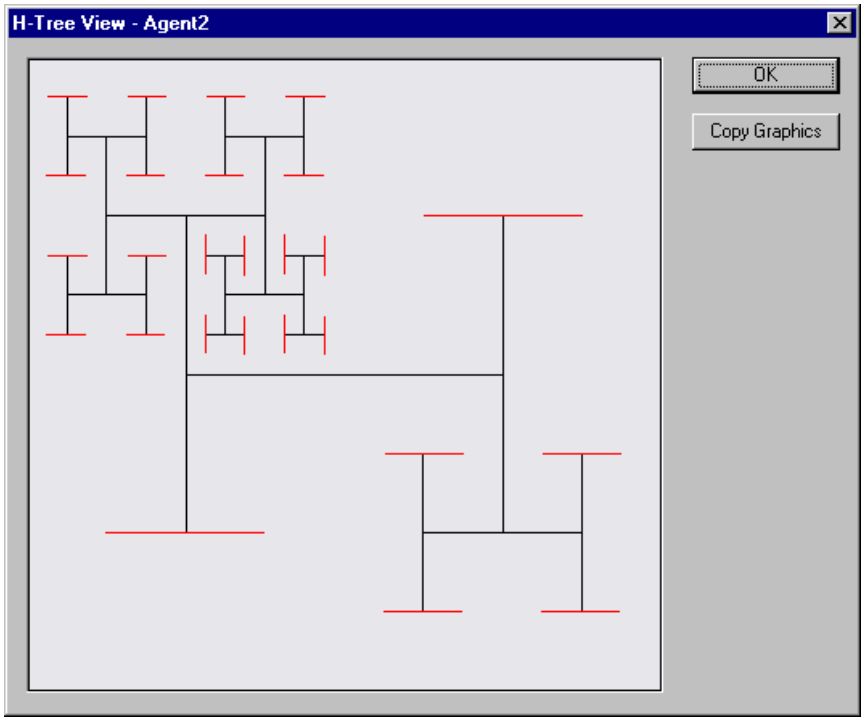


Figure 3 : Fenêtre « H-Tree View » de FRIENDS Offline.

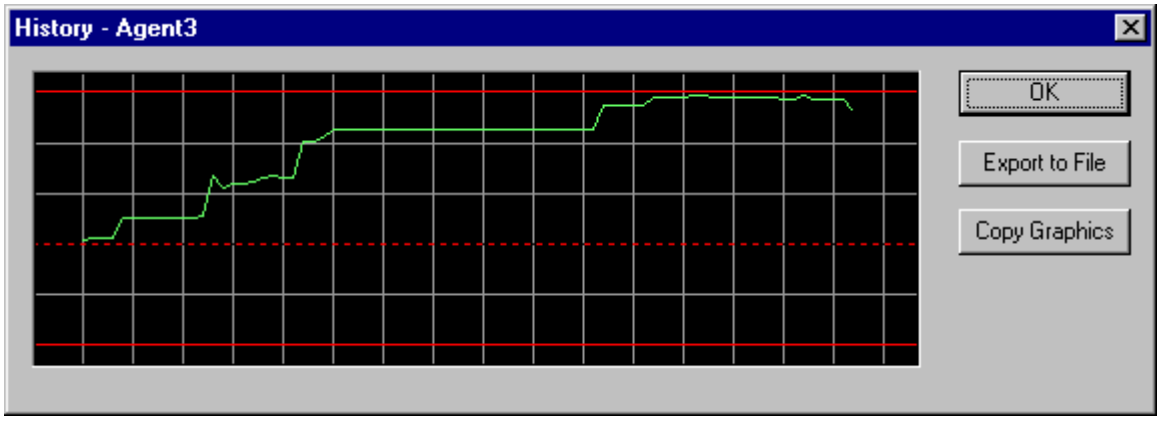


Figure 4 : Fenêtre « History » de FRIENDS Offline.

Pour assurer une expérimentation aisée, nous avons développé avec beaucoup de soin l'interface graphique de l'application. Ces efforts se sont avérés très profitables. L'interface graphique permet également, d'une manière conviviale, une présentation didactique des SMAM.

Un prototype de FRIENDS

FRIENDS Offline permet aux utilisateurs de créer des SMAM de plusieurs types différents, entre autre des SMAM linéaires et des agents atomiques d'ordre supérieur. Afin d'expérimenter le concept de FRIENDS, les utilisateurs peuvent également créer des SMAM *augmentés*.

En particulier, on peut créer des populations d'agents ayant des attributs étant des vecteurs trinaires en 5 dimensions. Quelques exemples de ce type de vecteur sont (0, 0, 0, 0, 0), (1, X, 0, 1, 1), et (X, X, X, 0, 1). Les symboles '1', '0' et 'X' représentent l'intérêt, le désintérêt, ou l'indifférence d'un utilisateur (représenté par un agent atomique) ou d'un groupe (représenté par un agent composé) pour une des cinq dimensions. Ces dimensions sont, en principe, abstraites, mais FRIENDS Offline les affiche, pour des raisons ergonomiques (ou didactiques), comme « POLITICS », « SCIENCE », « TRAVEL », « BUSINESS » et « ARTS ». Dans FRIENDS Offline, on peut déclencher l'affichage des attributs interprétés en plaçant le pointeur (la souris) au-dessus d'un agent augmenté. Au vecteur (0, 0, 0, 0, 0) correspond l'interprétation {« NOT interested in POLITICS », « NOT interested in SCIENCE », « NOT interested in TRAVEL », « NOT interested in BUSINESS », « NOT interested in ARTS »}. Le vecteur (1, X, 0, 1, 1) s'interprète comme {« interested in POLITICS », « NOT interested in TRAVEL », « interested in BUSINESS », « interested in ARTS »}. Finalement, (X, X, X, 0, 1) est interprété comme {« NOT interested in BUSINESS », « interested in ARTS »}. Notons que l'application attribue également un nom (tel que « Alain » ou « Lara ») à chaque agent atomique. Dans ce cas aussi, l'intérêt est principalement ergonomique (ou didactique).

Lorsqu'on crée un SMAM augmenté dans FRIENDS Offline, le nombre de dimensions des attributs est toujours 5. Toutefois, on est libre de choisir le nombre d'agents et la distribution des attributs. Les attributs peuvent être engendrés *soit* aléatoirement, *soit* étant identiques pour tous les agents, *soit* étant uniques (différents) pour tous les agents atomiques (jusqu'à 32 agents atomiques, ayant des attributs composés uniquement des symboles '1' et '0'). La dernière distribution est la plus intéressante, car elle peut générer ce que nous appelons, dans le contexte de FRIENDS, des populations parfaites. Formellement, nous définissons une **population parfaite** comme suit :

> Une population parfaite est une population qui peut être organisée de telle manière que chaque agent se trouve couplé à un agent différent dans une dimension seulement.

La figure 5 montre une population parfaite de taille 16. Répétons que, dans le contexte de FRIENDS, l'attribut d'un agent composé est égal à l'intersection des attributs des deux sous-agents.

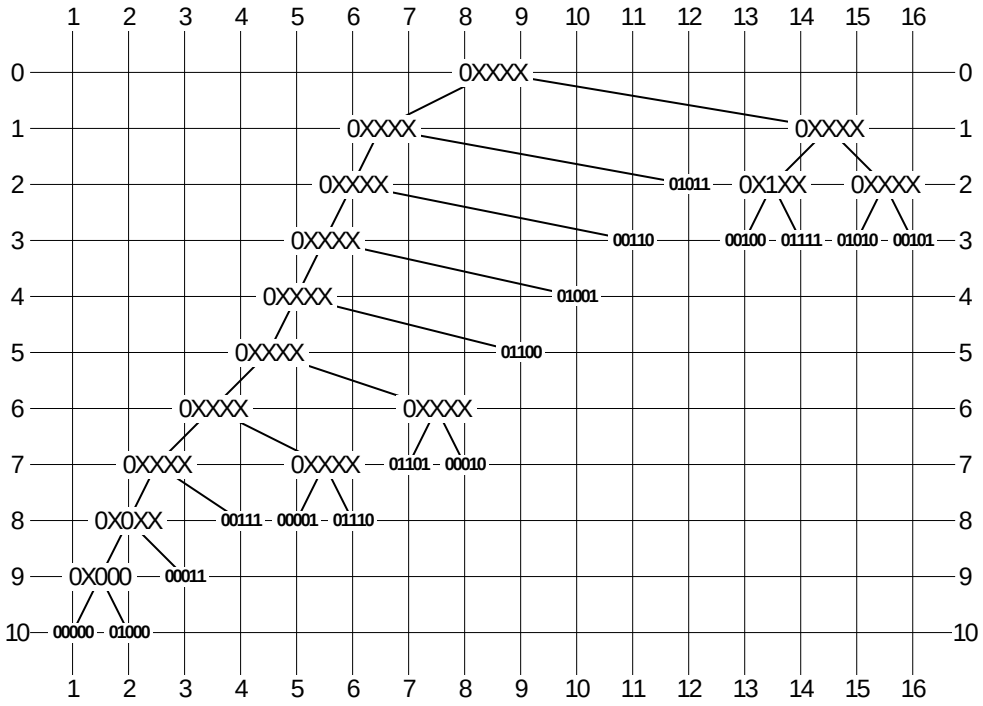


Figure 5 : Exemple d'une population parfaite.

La caractéristique principale d'une population parfaite consiste en le fait que son organisation optimale, tenant compte des valeurs des attributs, coïncide avec une organisation optimale selon la théorie des SMAM purs. Par exemple, le SMAM de la figure 5 peut être transformé en le SMAM de la figure 6. Dans ce SMAM transformé, tous les agents sont couplés avec des agents similaires, du point de vue « attributs » et du point de vue « structure ».

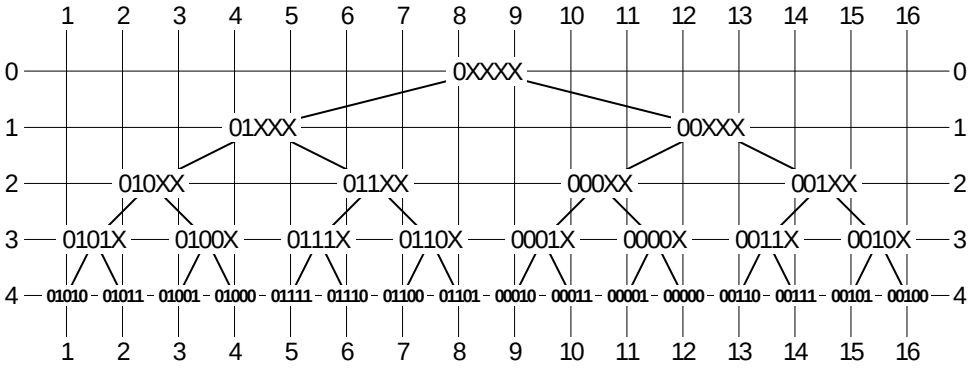


Figure 6 : Organisation optimale de la population de la figure 5.

Entropie et harmonie

L'équilibre d'un SMAM, calculé à partir de son entropie, nous donne une mesure du degré auquel - du point de vue structurel - les agents sont couplés à des agents similaires. Définissons également une mesure indiquant à quel degré les agents sont couplés à des agents ayant des attributs similaires. Les définitions suivantes sont spécifiques aux SMAM augmentés utilisés dans les versions différentes de « FRIENDS », membres de l'ensemble que nous appelons $\mathfrak{S}_{\text{FRIENDS}}$:

$\mathfrak{S}_{\text{FRIENDS}}$ = { S | S est un SMAM augmenté de type « FRIENDS » }

Définissons la **ressemblance** entre deux agents comme suit :

$\forall A, B \in \mathfrak{S}_{\text{FRIENDS}}$, la ressemblance $R(A, B)$ entre A et B est égal au nombre de mots-clés communs entre A et B.

Maintenant nous pouvons formuler l'**harmonie** d'un SMAM augmenté de type « FRIENDS ». Cette mesure, tenant compte des attributs des agents, joue le même rôle que l'entropie, tenant compte des structures des agents.

$\forall S \in \mathfrak{S}_{\text{FRIENDS}}$, l'harmonie $H(S)$ de S est donnée par :

$$H(S) = \sum_{(G\,D) \in S} R(G, D)$$

Le fait que, dans une population parfaite organisée de manière optimale, chaque agent est couplé à un agent similaire, différent dans une dimension seulement, implique qu'une partie maximale des attributs des agents peut être distillée et « promue » à des niveaux supérieurs. Ceci permet une navigation aisée à partir du SMAM complet : à chaque niveau, les deux sous-agents d'un agent composé, représentant des groupes ou des individus, peuvent être différenciés par leurs listes de mots-clés différentes. Dans le SMAM de la figure 5, par exemple, ceci n'est pas possible, car la plupart des agents composés ont comme attribut « 0XXX ».

De plus, nous pouvons démontrer que, dans une population parfaite, organisée de manière optimale, les navigations, par exemple du SMAM complet vers un agent atomique, sont - en moyenne - les plus courtes possibles. Cette propriété désirable peut être formulée comme la minimisation du nombre de navigations totales nécessaires pour accéder à tous les agents atomiques, donné par la formule suivante :

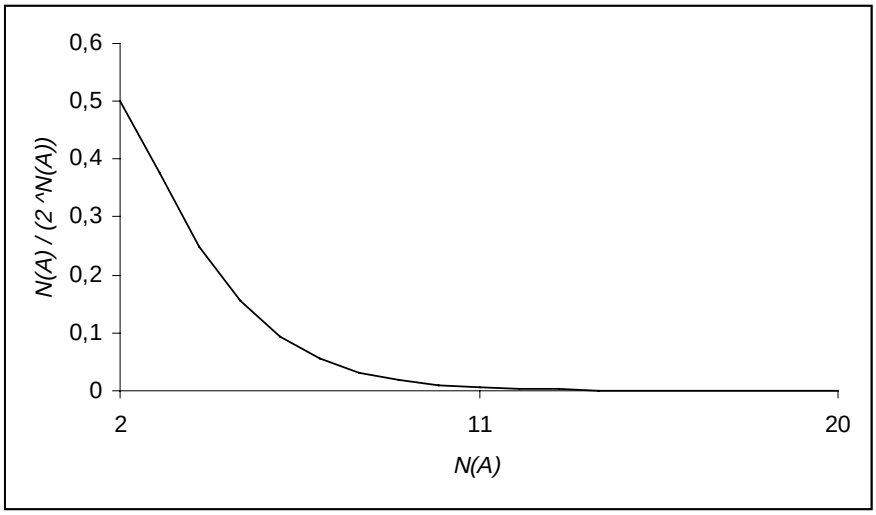
$$N_{total}(S) = \sum_{A \in S_\Delta} N(A)$$

A cause de la relation entre $N(A)$ et $N(A) / 2^{N(A)}$, illustrée par le graphique 1, la minimisation de la quantité ci-dessus est équivalente à la maximisation de la quantité ci-dessous, exprimant - en effet - l'entropie du système.

$$E(S) = \sum_{A \in S_\Delta} \frac{N(A)}{2^{N(A)}}$$

Donc, la maximisation de l'entropie, garantie par l'organisation optimale d'une population parfaite, implique que l'accès moyen aux agents atomiques est d'une longueur minimale.

Notons que, en général, plusieurs organisations optimales d'une population existent. La figure 7 montre un SMAM, différent du SMAM de la figure 6, mais représentant - lui aussi - une organisation optimale de la même population. Ceci ne pose aucun problème. En peut envisager d'ajouter des poids aux mots-clés afin de limiter le nombre de solutions possibles, mais nous n'avons pas exploré cette piste.



Graphique 1 : Relation entre $N(A)$ et $2^{N(A)}$.

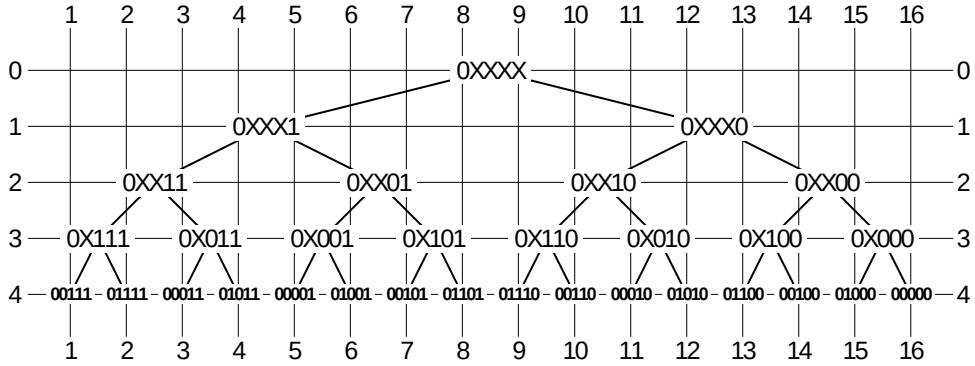


Figure 7 : Organisation optimale alternative de la population de la figure 5.

Comme nous l'avons expliqué, d'un point de vue applicatif, nous souhaitons, pour un système donné, une harmonie maximale et une entropie maximale. Ces deux conditions ne sont pas toujours compatibles : leur compatibilité dépend de la nature de la population. Comme nous l'avons vu, une population parfaite peut être complètement équilibrée. Par contre, le SMAM de la figure 8, représentant une population *presque* monotone, ne peut pas être équilibré mieux sans que son harmonie soit modifiée dans un sens négatif.

La sensibilité à la distribution des mots-clés sur la population est une limitation très importante de tout système effectuant le groupement d'utilisateurs en analysant les différences entre profils [Bothorel 98]. Toutefois, c'est notre hypothèse que des populations réelles évoluent - par les actions des individus modifiant leurs mots-clés -

vers des populations parfaites. Cette hypothèse, qui est périphérique aux travaux de cette thèse, est détaillée dans le chapitre IV.2, dédié à FRIENDS Online.

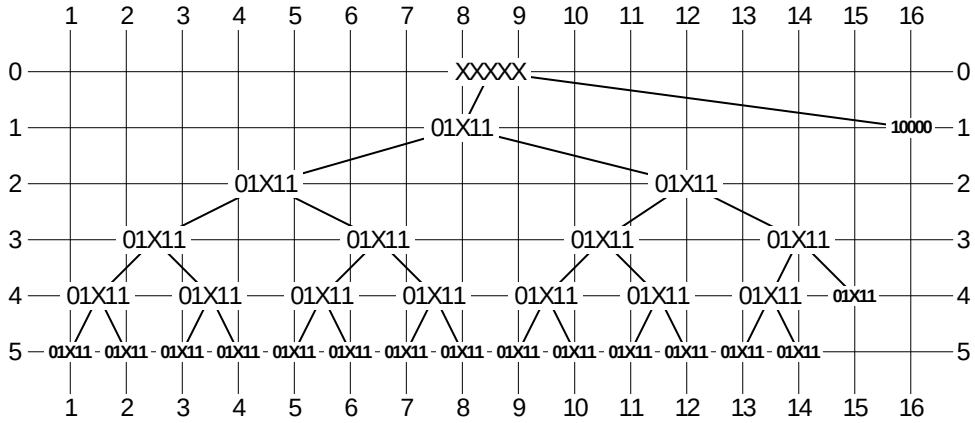


Figure 8 : Population *presque* monotone.

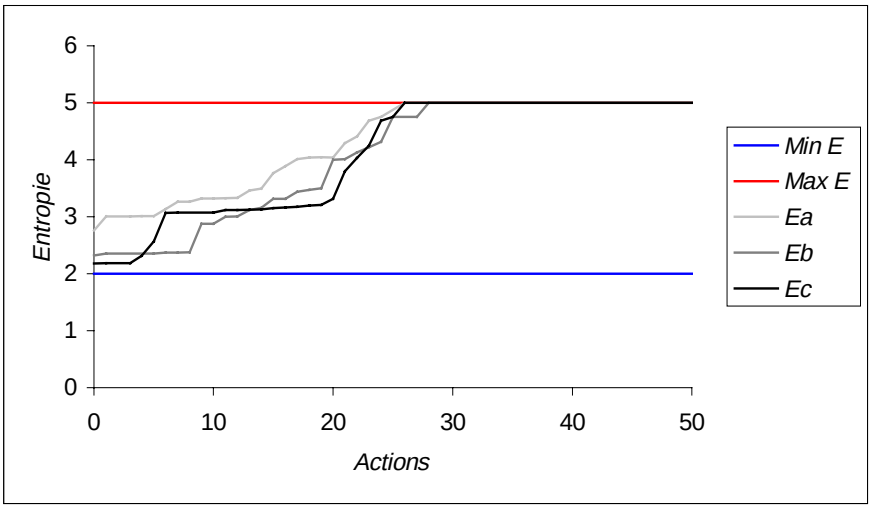
L'algorithme « global attributed III »

Nous avons expérimenté un nombre d'algorithmes implémentant le comportement « qui se ressemble, s'assemble » (en tenant compte des structures et des attributs). Ci-dessous, nous présentons un algorithme centralisé appelé « global attributed III », qui garantit l'évolution d'une population parfaite vers un état d'entropie et d'harmonie maximale. Les applications FRIENDS Numbercruncher et FRIENDS Online utilisent deux autres algorithmes élaborés dans les chapitres respectifs.

L'algorithme « global attributed III » est défini (pour *une* action) comme suit :

1) Créer une liste de tous les agents non couplés à un **ami**. *B* est un ami de *A* si tous les éléments de l'attribut de *A* sont identiques aux éléments de l'attribut de *B* jusqu'à une position *i* (non incluse) et si tous les éléments de l'attribut de *A* et tous les éléments de l'attribut de *B* sont 'X' à partir de la position *i* + 1.
2) Si la liste est vide, abandonner.
3) Trouver, dans la liste, le plus petit agent. S'il y en a plusieurs, en choisir un aléatoirement. Cet agent migrera.
4) Trouver, dans la liste complète de tous les agents, un **ami légal** de l'agent migrant. Un ami *B* de *A* est un ami légal si *B* ne fait pas partie de *A et* si *B* est différent du frère de *A et* si *B* est différent du père de *A*.
5) Faire migrer l'agent migrant vers son ami légal.

Le graphique 2 montre l'évolution de l'entropie de trois SMAM augmentés, guidée par l'algorithme « global attributed III ». Comme l'illustre le graphique, la convergence vers l'état optimal est rapide.



Graphique 2 : Evolution de l'entropie de trois SMAM augmentés.

III.2.2 L'implémentation de FRIENDS Offline

Afin de garantir une performance et une accessibilité maximale de notre application, nous l'avons développée en utilisant la plate-forme de développement Microsoft Visual C++ 5.0. A l'heure actuelle, C++ [Stroustrup 97] est un des langages orientés objet les plus performants. Par contre, C++ est peu utile sans une bibliothèque puissante de classes. Visual C++ nous offre les Microsoft Foundation Classes (MFC) permettant aux programmeur d'accéder aux fonctionnalités de Win32, les éléments de base de Windows 95/98 et Windows NT 4.0 [Prosise 96].

Visual C++ et MFC sont considérés comme des normes importantes dans la programmation de Windows : ils sont utilisés dans la réalisation d'applications telles que Microsoft Office 97, Adobe Photoshop 4.0 et Qualcomm Eudora Pro 3.0 [Visual C++]. A la base de la version 1.0 de Visual C++ était la septième version du compilateur C de Microsoft, à ce moment déjà connu pour son efficacité. Par conséquent, la plate-forme actuelle est très performante, même si elle est un peu moins sophistiquée comparée à certains environnements proposés par d'autres sociétés, plus orientés vers la programmation par composants.

Côté accessibilité, le choix de Visual C++ nous semble également raisonnable. Contrairement à des applications Java soigneusement écrites, notre application n'est pas indépendante du système de gestion. Toutefois, les plates-formes Windows 95/98 et Windows NT sont tellement répandues, même dans les laboratoires de recherches, que cette limitation nous semble peu gênante.

Exploiter la récursivité

Un des avantages d'un modèle récursif consiste en sa facilité d'implémentation sur un ordinateur numérique. Un grand nombre de méthodes écrites pour FRIENDS Offline sont d'une grande simplicité. Nous concluons ce chapitre en en présentant une petite sélection.

Les méthodes présentées sont toutes de la classe « CAgent », représentant les SMAM.

La méthode suivante retourne le maximum des niveaux occupés par les agents. La variable d'instance « rockBottom » indique si l'agent est un agent atomique d'ordre supérieur. Si c'est le cas, la variable d'instance « order » représente l'ordre de l'agent. Les pointeurs « pLeft » et « pRight » font référence aux deux sous-agents de l'agent. Le type « DWORD » représente des nombres naturels de 32 bits.

```
DWORD CAgent::Depth()
{
    if(rockBottom) {
          return order;
    } else {
          DWORD depthLeft = 1 + pLeft->Depth();
          DWORD depthRight = 1 + pRight->Depth();
          DWORD depth = depthLeft;
          if(depthRight > depthLeft) {
                depth = depthRight;
          }
          return depth;
    }
}
```

Les méthodes « RealSize( ) » et « Size( ) » retournent respectivement la taille réelle et la taille du SMAM. La fonction « power2( ) » retourne 2 à la puissance de son argument.

```
DWORD CAgent::RealSize()
{
    if(rockBottom) {
          return power2(order);
    } else {
          return pLeft->RealSize() + pRight->RealSize();
    }
}


DWORD CAgent::Size()
{
    if(rockBottom) {
          return power2(order + 1) - 1;
    } else {
          return 1 + pLeft->Size() + pRight->Size();
    }
}
```

La méthode « Entropy( ) » retourne l'entropie de l'agent, calculée à l'aide de la relation de récurrence présentée dans le chapitre II.2.

```
double CAgent::Entropy()
{
    if(rockBottom) {
          return double(order);
    } else {
          return 1.0 + (pLeft->Entropy() + pRight->Entropy()) / 2.0;
    }
}
```

Les deux dernières méthodes ajoutent, respectivement, tous les agents et tous les agents atomiques d'ordre supérieur à une liste (qui peut être vide avant l'appel de la méthode).

```
void CAgent::ListAgents(CList<CAgent*, CAgent*>* pList)
{
    pList->AddTail(this);
    if(!rockBottom) {
          pLeft->ListAgents(pList);
          pRight->ListAgents(pList);
    }
}


void CAgent::ListRockBottomAgents(CList<CAgent*, CAgent*>* pList)
{
    if(rockBottom) {
          pList->AddTail(this);
    } else {
          pLeft->ListRockBottomAgents(pList);
          pRight->ListRockBottomAgents(pList);
    }
}
```

CHAPITRE III.3

LES AGENTS MOBILES

The nice thing about standards is that there are so many to choose from.
Andrew S. Tanenbaum

III.3.1 La fonctionnalité des agents mobiles

A partir de 1995, à travers les travaux de Jim White et ses collègues à General Magic [White 95], le monde des agents a été enrichi d'un nouveau type : l'**agent mobile**. Un agent mobile est un agent capable de se déplacer d'un site physique à un autre. Dès sa naissance, le concept n'a pas arrêté de susciter l'intérêt des chercheurs et des ingénieurs. L'exploration du concept a été sensiblement aidée par le développement de Java, un environnement facilitant beaucoup le côté technique de la mobilité. Ces dernières années, deux séminaires internationaux ont étés organisés [Rothermel 97] [Rothermel 98] et un troisième, « Mobile Agents '99 », est programmé pour l'année 1999 [MA '99]. Toutefois, sur le plan commercial, le concept n'est pas encore adopté. Comme nous le verrons, ceci est peu surprenant car - pour l'instant - cette nouvelle technologie n'offre que peu d'avantages par rapport aux technologies classiques. Toutefois, et ceci est le sujet de cette section, nous croyons que les agents mobiles joueront un rôle crucial dans l'avenir de l'Internet et de l'informatique en général.

En particulier, nous présenterons notre opinion que la notion d'agent mobile est directement liée à celle des Systèmes Multi-Agents dotés d'une organisation dynamique. Donc, selon notre point de vue, elle est directement liée au modèle des SMAM et au concept de FRIENDS. Cette correspondance est la raison principale pour laquelle nous avons dédié un chapitre entier à la notion d'agents mobiles.

La mobilité

Détaillons d'abord le concept et étudions ses avantages comme ils sont présentées par ses partisans. Situons la notion en distinguant cinq éléments informatiques différents : les données, le code, les objets, les processus et les agents. Dans un système informatique composé de plusieurs ordinateurs, ces éléments peuvent être statiques ou mobiles. Donc, nous pouvons identifier dix types différents. Le tableau 1 montre des exemples de tous ces types. Nous avons choisi des exemples relativement familiers et concrets. Remarquons que la norme « CORBA » ne spécifie pas complètement les détails de *migrating objects*. Notons également que le système « Sprite » est relativement âgé. Nous l'avons cité à cause de sa réputation. Plus d'informations sur HTML, Java, CORBA, « Sprite » et les Aglets peuvent être trouvées respectivement à [HTML 4.0], [Java] et [OMG], et dans [Ousterhout 88] et [Lange 98a].

Il existe une vaste littérature dédiée aux éléments du tableau 1, même aux éléments mobiles seuls. Mais, dans le cadre de cette thèse, nous nous concentrons sur les agents mobiles dont l'exemple est mis en gras dans le tableau.

	Données	**Code**	**Objet**	**Processus**	**Agent**
Statique	un fichier « Word »	le code du programme « Word »	un objet manipulé par un processus exécutant « Word »	un processus exécutant « Word »	un assistant personnel
Mobile	un document HTML	un applet « Java »	un *migrating object* « CORBA »	un processus « Sprite »	**un Aglet implémentant un assistant personnel**

Tableau 1 : Eléments informatiques, statiques et mobiles.

Notons que la différence entre un processus et un agent n'est pas de nature technique, mais de nature conceptuelle. En effet, un agent est un processus ordinaire, *mais* doté - par l'utilisateur - d'une certaine autonomie. Toutefois, la conception par agent est tellement particulière qu'elle peut interagir avec le niveau technique. Plus loin dans cette section, nous détaillerons l'idée que le niveau conceptuel et le niveau technique ne sont pas indépendants et la question des agents mobiles n'est pas seulement une question d'implémentation. Dans un premier temps, continuons notre discussion du concept de base.

Les figures 1 à 5, ordonnées chronologiquement, illustrent le fonctionnement de base des agents mobiles.



Figure 1 : Migration (phase 1).

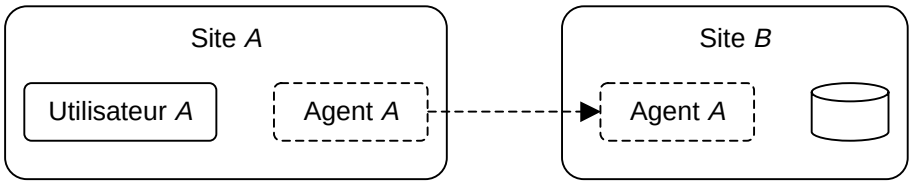


Figure 2 : Migration (phase 2).

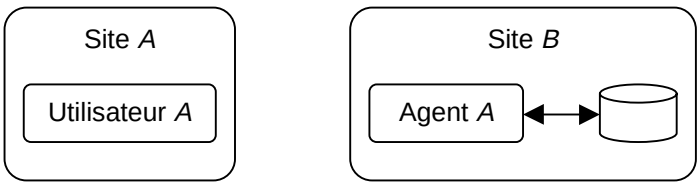


Figure 3 : Migration (phase 3).

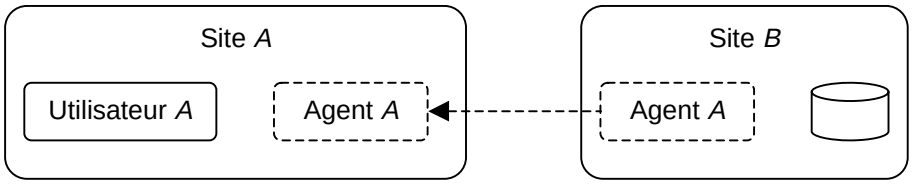


Figure 4 : Migration (phase 4).



Figure 5 : Migration (phase 5).

Supposons que l'utilisateur *A* veuille récupérer des informations sur un certain sujet. En interagissant avec son agent personnel, la requête est formellement spécifiée. Ceci est illustré par la figure 1. Puis, comme le montre la figure 2, l'agent migre de site *A* à site *B* sur laquelle se trouve une base de données contenant les informations cherchées. La figure 3 montre l'agent interagissant avec la base de données. Une fois l'agent satisfait, il retourne vers le site d'origine. Ceci est illustré par la figure 4. Finalement, comme le montre la figure 5, l'agent rapporte - d'une manière interactive - les résultats à son utilisateur.

La figure 6 montre l'alternative classique basée sur l'utilisation d'un agent statique : l'interaction avec la base de données est à travers du réseau.

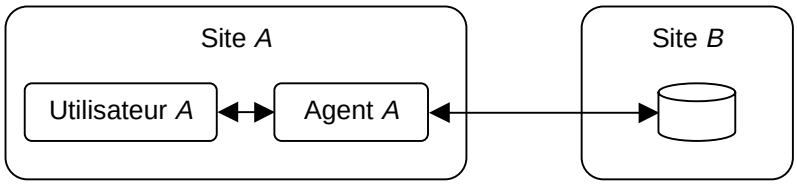


Figure 6 : Interaction à travers le réseau.

Ce qui distingue les deux approches est le fait que, dans la première, *les migrations exceptées*, il n'y pas de communication inter-sites. Etudions un deuxième exemple, encore plus parlant.

Trois utilisateurs souhaitent négocier, à travers de leurs agents personnels, l'heure d'un rendez-vous. La figure 7 montre une première approche à ce problème, basée sur le concept des agents mobiles. Dans cette approche, les trois agents se retrouvent dans un site commun où ils négocient. Une fois la négociation conclue, ils retournent chez eux pour rapporter le résultat. Notons que, afin de préserver de l'espace, nous avons visualisé les cinq étapes différentes de l'exemple en une seule figure.

La figure 8 montre l'approche classique, dans laquelle les agents interagissent à travers le réseau. Une fois de plus, la différence principale entre les deux approches est que, *si nous faisons abstraction des migrations*, des communications inter-sites peuvent être évitées dans l'approche basée sur l'utilisation des agents mobiles.

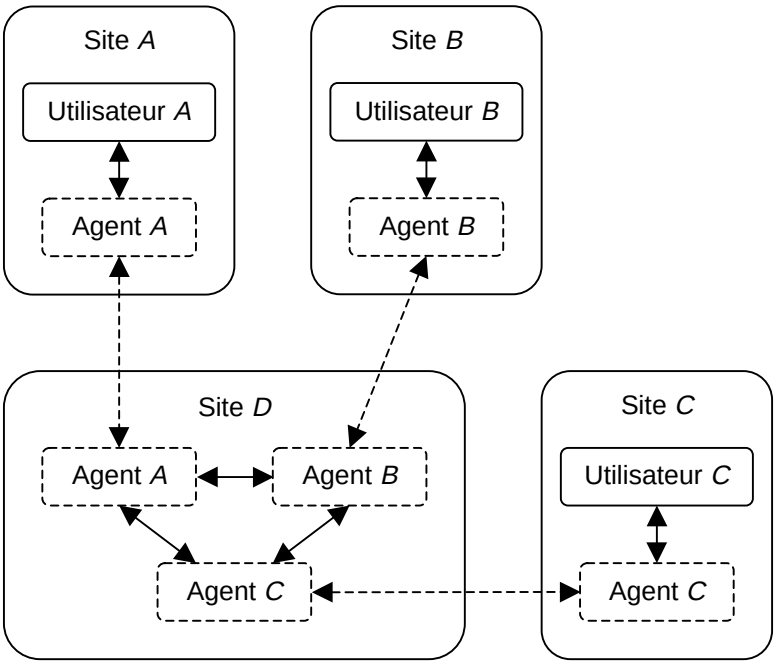


Figure 7 : Négociation entre agents mobiles.

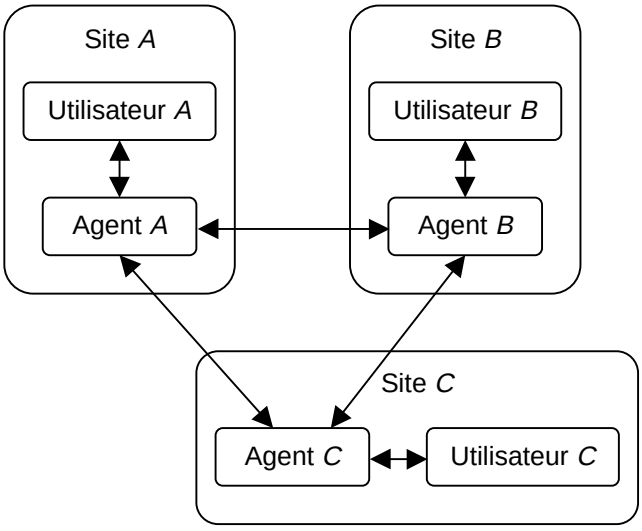


Figure 8 : Négociation entre agents statiques.

Exploiter la bande passante

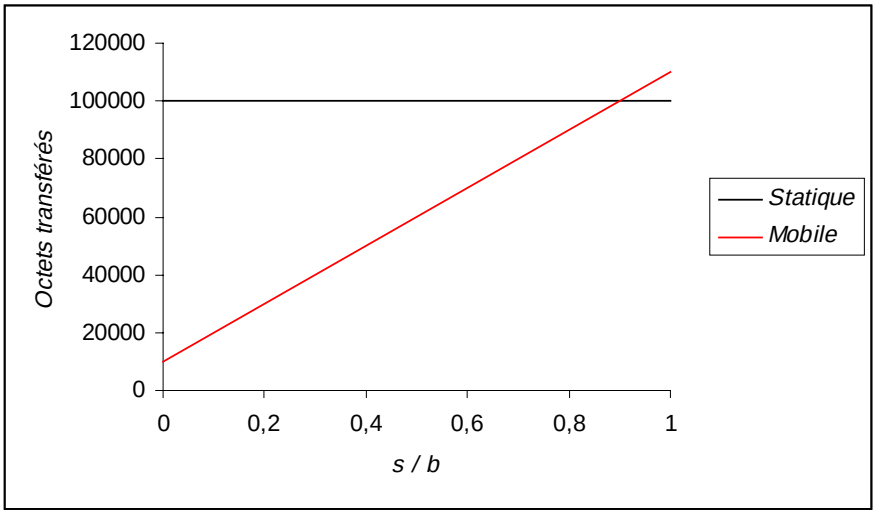
Dans un système informatique, les interactions intra-site sont moins coûteuses que les interactions inter-sites. Par conséquent, dans le contexte des deux exemples présentés, les systèmes utilisant les agents mobiles semblent être plus efficaces que les systèmes classiques. La réduction de l'utilisation de bande passante, par l'application des agents mobiles, est l'argument principal présenté par ses avocats [Chess 95] [Chess 97] [Harrison 95] [White 95].

L'argument semble solide, mais l'est-il vraiment ? En effet, quelle est l'importance du coût des migrations ? Etudions cette question en reprenant le premier exemple.

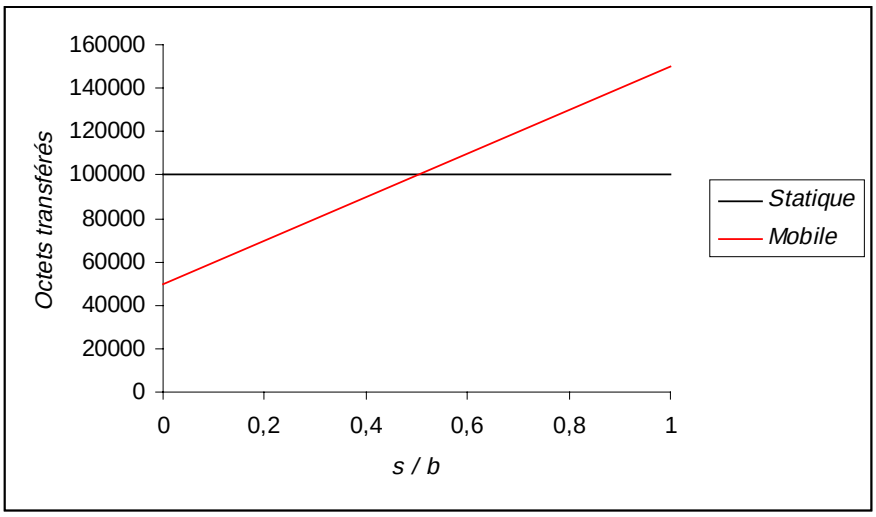
Dans la première phase de l'approche « agents mobiles » de cet exemple, l'utilisateur et l'agent établissent la requête. Supposons qu'elle ait une taille de *r* octets. Puis l'agent migre vers le site *B*. Supposons que la taille de l'agent soit fixe à *a* octets. L'agent interagit avec la base de données dont la taille est de *b* octets. Finalement, l'agent rentre pour communiquer la solution composée de *s* octets. Donc, la quantité totale de données transférées entre les sites différents est (*a* + *r*) + (*a* + *s*) octets. Notons qu'il n'est pas toujours nécessaire qu'un agent retourne chez son utilisateur et que - souvent - il suffit qu'il envoie le résultat. Toutefois, en général, on suppose que l'agent rentre.

Dans l'approche classique de cet exemple, utilisant un agent statique, la quantité totale de données transférées peut monter jusqu'à *r* + *b* octets. En effet, dans les pires des cas, on demande le transfert de la base complète et on la traite localement.

Comparons les deux approches, en supposant que *r* est négligeable. Le graphique 1 montre le nombre d'octets transférés, dans les deux approches, pour *b* = 100000, *a* = 5000 et *s* / *b* entre 0 et 1. Le graphique 2 visualise la même relation, mais pour *a* = 25000.



Graphique 1 : Agents statiques vs. agents mobiles (*a* = 5000).



Graphique 2 : Agents statiques vs. agents mobiles (*a* = 25000).

Comme le montrent les deux graphiques, l'application des agents mobiles n'est pas toujours avantageuse. Surtout lorsque les tailles des agents et des solutions (relative à la base de données) sont grandes, l'approche classique est préférable. Cependant, en général, la taille des agents et le ratio *s* / *b* sont modestes, et la mobilité prometteuse.

Par contre, ce raisonnement n'est valide que si la requête ne peut pas être formulée d'une telle manière qu'elle peut être traitée directement pas le système de gestion de la base de donnée. Dans ce cas, il suffit d'envoyer la requête et de recevoir la solution sans impliquer des agents. Cette solution est toujours la plus économique car seulement *r* + *s* octets doivent être transférés. Donc, cette application très étudiée des agents mobiles n'apporte aucun avantage si les requêtes peuvent être normalisées pour les bases de données impliquées. Ceci est sans doute une des raisons principales pour laquelle les agents mobiles ne sont pas encore utilisés à une échelle importante.

Toutefois, dans un monde informatique de plus en plus sophistiqué et personnalisé, nous pouvons nous attendre à une croissance importante de requêtes non standards. De plus, il est possible que certains utilisateurs préféreront envoyer leurs agents personnels plutôt que faire confiance au système de gestion distant, en estimant - par exemple - que leurs agents sont plus fiables.

L'importance de la mobilité

Evidemment, comme l'illustre le deuxième exemple que nous avons donné, les agents mobiles peuvent être appliqués à d'autres problèmes que la consultation de bases de données. Lorsqu'il y a des interactions intenses entre plusieurs agents, il est intéressant de les grouper ensemble. Si le nombre d'agents est important, par exemple dans un système sur l'Internet, représentant les utilisateurs par des agents, la distribution intelligente des agents n'est pas seulement avantageuse, mais simplement indispensable. En effet, à cause de sa grande taille, un tel système doit forcement être distribué et les interactions inter-sites doivent être limitées à un minimum. Il est également important que ces systèmes permettent, au niveau conceptuel, une distribution naturelles des agents selon l'intensité des interactions. Sinon, les implémentations de ces systèmes s'avéreront difficiles, et des problèmes de *scaling* inévitables.

Sachant la nature des environnements tels que l'Internet, on peut s'attendre à une grande dynamique de ces systèmes, nécessitant une organisation flexible des agents. Au niveau conceptuel, ceci implique l'organisation dynamique des agents. Au niveau de l'implémentation, ceci implique l'utilisation d'agents mobiles suivant les changements organisationnels au niveau conceptuel. Sinon, comme nous l'avons indiqué ci-dessus, on sera confronté aux problèmes de *scaling*. Donc, la nature même de systèmes de très grande taille, comme ceux utilisés sur l'Internet, nécessite l'organisation dynamique des agents, l'application d'agents mobiles et une correspondance continue entre ces deux niveaux, l'un conceptuel, l'autre technique. Dans ce contexte de l'Internet, il est incorrect de considérer les agents mobiles comme un détail technique.

Dans leur *column* dédié aux agents dans la magazine *Internet Computing*, Michael Huhns et Munindar Singh expriment une certaine réservation à propos des agents mobiles : « …anything that can be done with mobile agents can also be done with conventional software techniques… » et « Only rarely do circumstances call for an agent to be mobile, in spite of all the effort being spent on developing techniques for mobility. » [Huhns 97]. Comme nous l'avons expliqué, il est vrai que - dans un grand nombre de circonstances - les agents mobiles ne

sont pas utiles. Toutefois, nous répétons que le concept et la technique nous permettent de construire des systèmes de grande taille, naturellement distribuée. Huhns et Singh semblent argumenter que la distribution d'un Système Multi-Agents peut facilement être déléguée au niveau technique et que ce niveau ne doit pas interférer avec le niveau conceptuel. Considérant l'histoire de l'informatique, nous ne pouvons pas partager cette vue.

Ironiquement, les auteurs eux-mêmes illustrent la nature irréaliste de leur vue. Ils argumentent que les agents mobiles constituent une approche *procédurale*, plutôt qu'une approche *déclarative*. Puis, ils argumentent que, dans l'informatique, les systèmes déclaratifs ont toujours remplacé les systèmes procéduraux. Ceci n'est pas correct.

Pendant l'histoire de l'informatique, un grand nombre de langages déclaratifs ont été introduit. Lisp et Prolog ne sont que deux exemples. Toutefois, après toutes ces années, virtuellement tous les programmeurs utilisent des langages procéduraux tels que C, C++ ou Java. Lisp existe toujours, mais plutôt utilisé comme langage procédural. Dans leur article, Huhns et Singh représentent l'école argumentant qu'il vaut mieux spécifier le « quoi » que le « comment ». Toutefois, cette approche ne marche que dans des cas très spécifiques et souvent académiques. L'informatique n'est pas les mathématiques et les programmeurs le savent. Lisp (si utilisé d'une manière déclarative) et Prolog sont des langages d'une grande élégance, mais peu adaptés à la construction des systèmes interactifs d'aujourd'hui. Il ne s'agit pas d'une question de performance, comme l'illustre le succès de Java, relativement peu performant, mais d'une question de modélisation.

Répétons que - selon notre point de vue - le concept des agents mobiles est important car il nous permet de lier, d'une manière naturelle, le niveau conceptuel d'un SMA doté d'une organisation dynamique à son implémentation distribué. Le système « FRIENDS », introduit dans les chapitres précédents est un exemple excellent d'un tel SMA, nécessitant une implémentation distribuée. Puisque l'organisation dynamique des agents joue un rôle crucial dans son fonctionnement, elle est bien supportée. De plus, à cause de la nature binaire et récursive des SMAM, un système « FRIENDS » peut être naturellement et presque sans contraints être divisé en plusieurs sous-systèmes résidants à des sites différents. Nous reprendrons cette problématique dans le chapitre IV.2.

Concluons cette sous-section en soulignant que l'interaction directe avec des services ou avec d'autres agents n'est pas le seul avantage que peut offrir la migration. En effet, le but principal d'un agent vraiment autonome n'est pas seulement d'exécuter des tâches, mais également de survivre. Pour survivre, un agent a besoin de ressources informatiques, et en première place de temps de calcul. Lorsque les ressources d'un site donné sont complètement utilisées, un agent résidant sur ce site risque d'être sérialisé sur disque. Une fois la sérialisation effectuée, l'agent a perdu complètement son autonomie et doit compter sur d'autres entités pour être « réanimé ». Cette situation est, du point de vue de l'agent, de son propriétaire et de ses constructeurs absolument à éviter.

Afin d'éviter la sérialisation, les agents « malins » migreront vers des sites moins chargés, une fois la charge de leur site actuel devenue trop grande. Ce type de migration existera parallèlement à la migration pour des raisons organisationnelles. Notons que, à cause de cette nécessité, nous pouvons nous attendre, à une échelle mondiale, à une migration continue d'agents d'est en ouest. En effet, à tout moment donné, la plupart des ressources informatiques sont disponibles là où dorment leurs propriétaires.

La mobilité et la migration

Nous avons utilisé les termes « migrer » et « migration » dans deux contextes différents. A un niveau conceptuel, un agent migre lorsqu'il change de place dans l'organisation d'un SMAM. A un niveau technique, un agent migre lorsqu'il se déplace de site à site. En général, le sens exact du terme doit être déduit du contexte de son utilisation. Lorsque le danger de confusion est réel, nous distinguons les deux concepts en utilisant les termes « migration conceptuelle » et « migration physique ».

Nous pourrions réserver les termes « migrer » et « migration » pour la migration conceptuelle seule. Mais, ce vocabulaire est déjà utilisé par les chercheurs étudiant la migration physique. De plus, dans d'autres domaines scientifiques ces termes sont généralement utilisés dans le sens physique (par exemple, « la migration de particules » en Physique, « la migration d'oiseaux » en Biologie, « la migration de planètes » en Astronomie, etc.).

Par contre, lorsque nous parlons de la mobilité, nous référons exclusivement à la migration physique. Donc, selon notre terminologie, les agents mobiles sont des agents qui migrent *physiquement*. Notons que la mobilité est une capacité, plutôt qu'une action, comme la migration. Si nous voulons utiliser les deux termes au même niveau, nous devons parler de « mobilité » et de « capacité de migration (virtuelle ou physique) ».

En théorie, la capacité de migration virtuelle et la mobilité sont deux concepts indépendants. Toutefois, en pratique, comme nous l'avons expliqué ci-dessus, ils peuvent être très liés.

III.3.2 Implémenter la mobilité

Les mécanismes à la base de la technologie des agents mobiles sont relativement simples. Un agent est typiquement composé de **code**, **données** (état au niveau conceptuel), et **état d'exécution** (état au niveau de l'implémentation). Une implémentation complète de la mobilité assure, lorsqu'un agent migre, le transfert de tous ces trois éléments. Un bon exemple d'un tel système est Telescript, développé par General Magic [White 95].

Les mécanismes de base

La figure 9 montre, du point de vue de l'utilisateur, un exemple illustrant les éléments essentiels d'un système « Telescript ». Nous y trouvons deux sites physiques différents, un utilisateur, une « base de données » représentant un service utile à l'utilisateur, et six agents. En plus de ces éléments, la figure montre trois **lieux** (*places* dans la terminologie de Telescript). En effet, les agents ne se trouvent pas comme des processus complètement indépendants dans les sites : ils sont contenus dans des lieux assurant et contrôlant les interactions entre les agents, les services et les ressources des sites. L'utilisation de lieux ne facilite pas seulement la gestion des agents, mais offre également un mécanisme pour garantir la sécurité du système.

Les questions de sécurité jouent un rôle essentiel dans la problématique des agents mobiles. En effet, plus le nombre de sites accueillant des agents est grand, plus le concept de la mobilité est utile, mais les propriétaires des sites n'accepteront jamais l'immigration d'agents inconnus si l'intégrité de leur système n'est pas garantie. Donc, il est peu surprenant qu'un nombre important de chercheurs dans la communauté des Agents Mobiles étudient le problème de la sécurité (voir, par exemple, [Karjoth 97] [Vigna 98] [Walsh 98]). Ils étudient non

seulement comment on peut protéger les sites contre des actions nocives de la part des agents, mais également comment les agents eux-mêmes peuvent être protégés contre les actions des sites et leurs cohabitants.

La figure 9 montre l'agent *E* migrant entre les sites *A* et *B*. Pour qu'il puisse migrer, son code, ses données et son état d'exécution doivent être sérialisés et transférés par le réseau. Puis, dans l'autre site, l'agent doit être reconstitué sur la base de ces trois éléments sérialisés. Le transfert même des éléments ne pose aucun problème : il ne s'agit d'un simple application de la technologie des réseaux. Toutefois, la sérialisation et la reconstitution sont plus délicates.

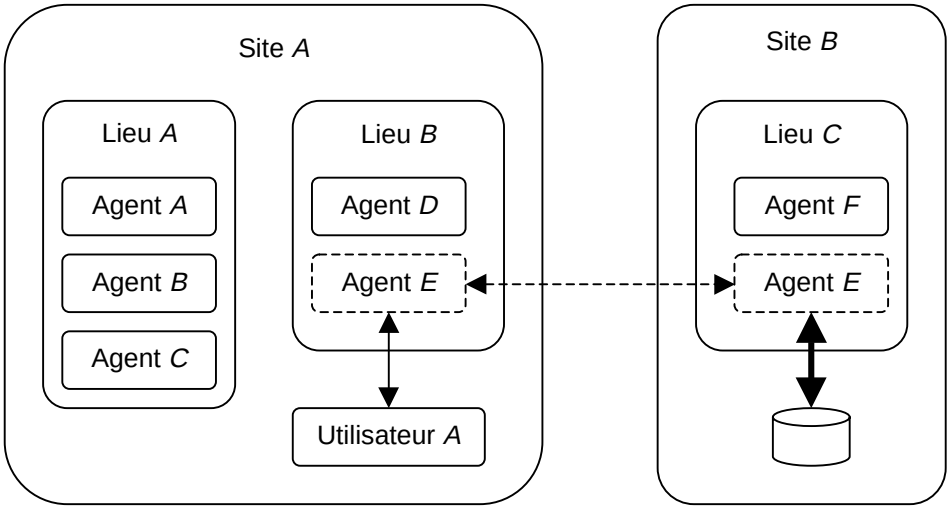


Figure 9 : Eléments essentiels d'un système « Telescript ».

Sérialiser et reconstituer les données est relativement facile. La procédure peut être plus complexe si les données sont hautement structurées. Heureusement, les langages modernes, tels que Java, supportent des mécanismes de sérialisation, facilitant sensiblement le processus.

La sérialisation et la reconstitution de code étaient, jusqu'il y a quelques années, plus ésotériques, mais l'apparition de Java nous a tous familiarisé avec ce mécanisme. Les applets ne constituent peut-être pas l'intérêt principal de ce jeune langage, mais elles ont motivé toute une communauté à étudier le concept. En utilisant Java, comme nous le monterons dans la sous-section suivante, il est facile de faire migrer le code d'un agent mobile.

Sérialiser et reconstituer l'état d'exécution d'un processus reste un problème difficile. Cette fonctionnalité était supportée par Telescript, mais peu de systèmes actuels la supportent. Une des raisons principales de cette absence est le fait que Java ne supporte pas, pour des raisons de sécurité, la sérialisation de cet état, par exemple en permettant la sérialisation de *threads*. Toutefois, un certain nombre de chercheurs développent des systèmes capables de la sérialisation et reconstitution de l'état d'exécution, entre autre en modifiant la machine virtuelle de Java [Fünfrocken 98]. En ce qui concerne les agents, l'absence de cette fonctionnalité ne nous semble pas très gênante. Elle implique que, après chaque migration, l'agent commence son exécution à partir d'un point d'entré fixe, par exemple par l'exécution de la méthode « Arrivé( ) ». Puisque nous sommes convaincus que la migration doit constituer un événement important, plutôt que transparent, dans la vie des agents, cette restriction ne nous gène pas. Evidemment, dans un contexte de processus migrant plus général, la migration est sensée d'être

transparente. Soulignons cependant que, dans ce contexte, différent du nôtre, le terme « agent mobile » ne devrait pas être utilisé.

Dans un système concret, les agents ne peuvent pas se faire migrer eux-mêmes : ils ont besoin d'une entité externe s'occupant de ce processus. Cette entité est typiquement appelée le **moteur de migration** (*engine* dans la terminologie de Telescript) et doit être installée sur tous les sites entre lesquels les agents souhaitent migrer. Ce moteur, gérant les lieux, peut être complètement indépendant. Toutefois, il peut coopérer avec ces autres systèmes, qui peuvent être intégrés dans le système de gestion. Les figure 10 à 13 illustrent quatre cas différents. Dans ces figures, nous avons fait abstraction des lieux. Notons que d'autres configurations, plus exotiques, peuvent être utilisées.

La figure 10 illustre la configuration la plus simple. Un seul système de gestion est supporté et une seule version du moteur est nécessaire. L'agent peur s'exécuter directement au-dessus du système de gestion, mais sous le contrôle du moteur. Le moteur effectue également les migrations.

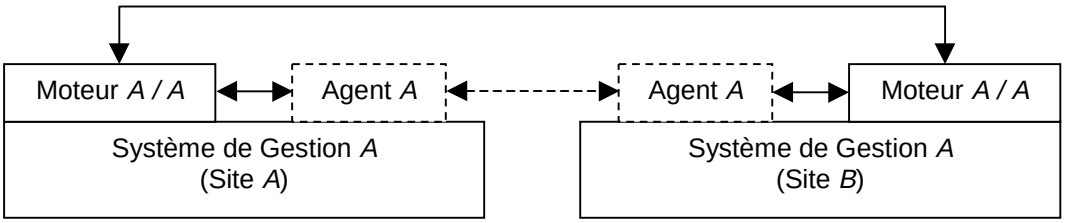


Figure 10 : Implémentation de la mobilité (cas 1).

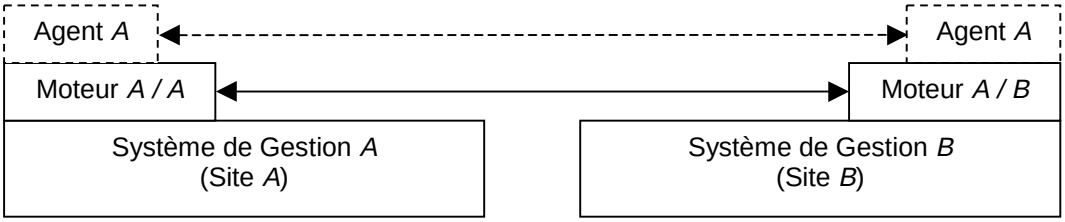


Figure 11 : Implémentation de la mobilité (cas 2).

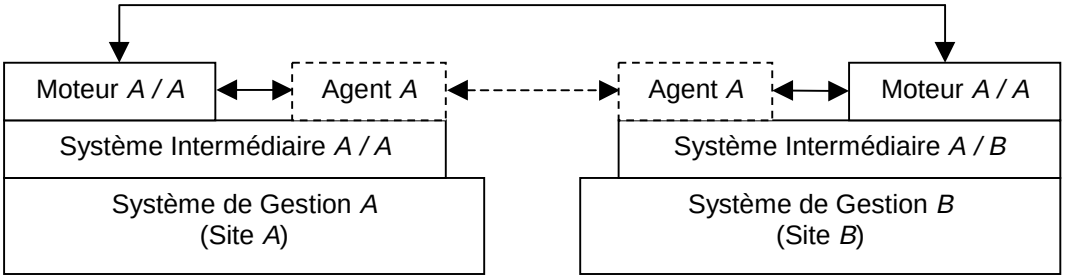


Figure 12 : Implémentation de la mobilité (cas 3).

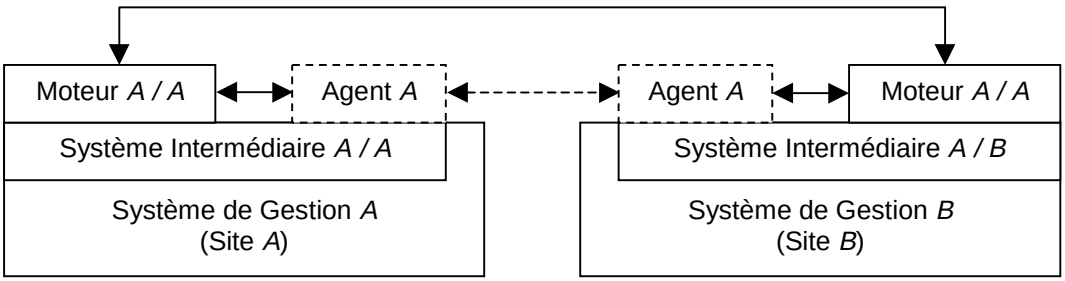


Figure 13 : Implémentation de la mobilité (cas 4).

Le système de la figure 11 supporte plusieurs plates-formes et nécessite plusieurs versions du moteur, une pour chaque plate-forme. De plus, les agents ne peuvent pas être exécutés immédiatement au-dessus du système de gestion, mais doivent être interprétés par les moteurs.

La figure 12 montre encore une autre solution permettant les agents de migrer entre sites équipés de systèmes de gestion différents. Elle exploite un système intermédiaire cachant aux moteurs et aux agents le système de gestion utilisé. Java est un bon exemple d'un tel système intermédiaire, dont la machine virtuelle interprète les moteurs et les agents. Evidemment, l'interopérabilité n'est possible que lorsque le système intermédiaire est supporté sur les plates-formes différentes.

La figure 13 montre le même système que la figure 12, mais avec le système intermédiaire intégré dans le système de gestion. Notons que Java, par exemple, est de plus en plus intégré dans les systèmes de gestion actuels.

Illustrons maintenant comment la migration du code d'un agent mobile peut être implémentée en utilisant Java. Le mécanisme illustré est typiquement utilisé dans les systèmes des figures 12 et 13. Notons que nous nous concentrons sur la migration du code, car la migration des données de l'agent est relativement triviale à implémenter. Répétons que, en utilisant Java, le transfert de l'état d'exécution est - en principe - impossible.

Le « classloader » en Java

Quand nous avons commencé les travaux à la base de cette thèse, Java n'était pas encore un langage bien connu. Toutefois, il offrait déjà la plupart des avantages qu'elle offre maintenant : orienté objet, simple, indépendant de la plate-forme et incluant une bibliothèque riche de classes, dont un grand nombre orientées vers l'Internet [Gosling 95]. Des informations sur la version actuelle peuvent être trouvées à [Java] et dans [Arnold 98]. Il est vrai que, comparé à C++, Java est moins performant et moins mature, mais dans certains contextes, tels que la programmation d'applications pour l'Internet, ses avantages dominent.

Nous avons construit notre première application « Java » en été 1995. A ce moment, la version la plus avancée du langage était la version *alpha* 3, supportée uniquement par le navigateur « HotJava » de Sun. Java n'était pas encore intégré dans les autres navigateurs. Son utilisation nous a permis de réaliser l'application, appelée « le Salon », très orientée vers l'Internet, dans un temps record [Van Aeken 96a]. En conséquence, nous avons réutilisé Java pour toutes nos applications Internet, FRIENDS Online inclus.

Avant d'implémenter FRIENDS Online, nous avons exploré la problématique des agents mobiles. Par exemple, nous avons construit notre propre système supportant un mécanisme de base nécessaire pour la migration

physique d'agents [Van Aeken 97]. Dans ce système, le moteur d'un site *A* demande au moteur d'un site *B* d'accueillir un agent résidant à *A*, en lui présentant l'URL du code de l'agent. Dans une deuxième phase, le moteur de *B* récupère le code, reconstitue l'agent et l'exécute. Notons qu'il s'agit d'un exemple spécifique ne nécessitant pas le transfert de données (de l'agent). Toutefois, cette fonctionnalité peut facilement être ajoutée. Par contre, le système n'exécute qu'un agent à la fois. Evidemment, un système plus sophistiqué doit supporter l'exécution (apparemment) parallèle des agents, en attribuant un ou plusieurs *threads* à chaque agent.

Etudions l'essentiel du code de ce simple système :

```
class Engine {

public static void main(String[] args) {

    //…

    NetworkClassLoader loader = new NetworkClassLoader();
    Class agentClass = null;



    try {
          String URLname = dis.readLine(); // Lire URL
          loader.setURL(URLname);
          agentClass = loader.loadClass("Agent", true);
    } catch(Exception e) {
          System.out.println("Problems while loading class.");
    }

    Agent agent = null;
    try {
          agent = (Agent)(agentClass.newInstance());
          agent.execute();
    } catch(java.lang.Exception e) {
          System.out.println("Problems while creating " +
                "or executing agent instance...");
    }

    //…

}
}
```

La classe « NetworkClassLoader » est une sous-classe relativement simple de la classe « ClassLoader », intégrée dans Java. Les mécanismes à la base de son implémentation sont détaillés dans des articles de Chuck McManis et de Jack Harich dans JavaWorld [JavaWorld].

Comme le montre le code ci-dessus, la récupération, restitution et exécution du code d'un agent sont presque triviales en Java. Il est donc peu surprenant que la plupart des systèmes actuels, supportant la mobilité, soient basés sur ce langage (et son environnement).

Le protocole de migration

La migration, indépendamment du fait qu'elle soit virtuelle ou physique, nécessite un protocole. Nous concluons cette section en en présentant un modèle générique. Dans le chapitre IV.2, nous détaillerons un protocole plus spécifique, celui utilisé dans FRIENDS Online MAI.

Dans la migration d'un agent *A* d'une source *S* à une destination *D*, nous pouvons identifier les trois étapes suivantes :

1. *A* décide de migrer ou *S* ou *D* demandent à *A* de migrer.
2. *A* demande à *S* la permission d'émigrer (si elle n'est pas encore accordée) et à *D* la permission d'immigrer (si elle n'est pas encore accordée). Cette étape peut impliquer des négociations et des vérifications au niveau de la sécurité et des ressources des deux systèmes.
3. Si les permissions de *S* et de *D* sont obtenues, *A* migrera de *S* à *D*. La migration nécessitera la réorganisation de *S* et de *D*.

Notons que la migration de *A* de *S* à *D* peut être vue, jusqu'à un certain degré, comme la combinaison de trois événements relativement indépendants : la mort de *A* à *S*, la naissance de *A* à *D* et le transfert neutre de *A* de *S* à *D*. Donc, en général, les processus de création et de destruction d'agents et les processus de migration peuvent partager un certain nombre de mécanismes.

III.3.3 Les systèmes existants

Plusieurs systèmes, commerciaux et autres, supportant la mobilité sont actuellement disponibles. Dans cette section, nous en présentons les plus importants. Notons que, à part de cette présentation, plusieurs études et comparaisons existent dans la littérature sur les agents (par exemple, [Cockayne], [Genovese 97] et [Kiniry 97]).

Des critères différents peuvent être utilisés pour valider et comparer ces systèmes. Un des critères principaux est la **distribution** (actuelle ou attendue) du système. En effet, à une échelle globale, même un système parfait ne sert à rien si son utilisation est limitée à un seul site : plus il y a des sites disponibles, plus la mobilité devient intéressante. De plus, un système largement distribué est typiquement plus mis à l'épreuve et, par conséquent, relativement libre de bogues et de problèmes de sécurité. La **stabilité** et la **sécurité** sont, en effet, deux autres critères très importants. L'**efficacité**, au niveau de l'exécution des agents et au niveau de l'utilisation du réseau, joue également un rôle crucial. La **facilité de conception** d'applications est un autre critère important. Un système incorporant des modèles SMA ou des protocoles de communication sophistiqués a plus d'intérêt pour nous qu'un système ne supportant que les fonctionnalités de base. A un niveau plus bas, **la facilité de programmation** joue également un rôle. Le **coût** des licences du système est un dernier critère que nous considérons.

Ci-dessous, nous présenterons - brièvement - les systèmes les plus répandus, suivi par une courte évaluation.

Agent Tcl (D'Agents)

Un des premiers systèmes supportant les agents mobiles a été développé par Robert Gray, David Kotz et d'autres à Darthmouth College [Agent Tcl]. A l'origine, le système n'avait pas de nom et utilisait TCP/IP, le courrier électronique et la commande UNIX « rshell » comme mécanismes de transfert [Kotay 94]. Plus tard, la même

équipe a introduit « Agent Tcl », un système utilisant Tcl comme langage principal de programmation des agents [Gray 95] [Kotz 97]. Toutefois, à la base, le système supporte plusieurs langages, à savoir Tcl, Java et Scheme, et récemment le système a été rebaptisé « D'Agents ». Notons qu'Agent TCL est un des seuls systèmes actuels n'étant pas basé directement sur Java.

Aglets

Le « Aglet Workbench » d'IBM est sans doute un des systèmes les plus connus [Aglets]. Danny Lange, un des pionniers de la communauté, était à la base de son développement au Japon [Lange 98a] [Lange 98b]. Le système est facile à charger du Web et à installer sur tout système supportant Java. Il est assez fiable, mais la documentation en ligne n'est pas d'une qualité commerciale. Notons que Danny Lange a quitté IBM pour joindre General Magic dans le développement d'Odyssey.

Concordia

Concordia est également un système basé sur Java, mais développé par le Horizon Systems Laboratory de Mitsubishi [Concordia] [Wong 97]. Il s'agit d'un système récent, orienté vers des applications au niveau des entreprises. Par conséquent, la sécurité du système est exceptionnellement soignée [Walsh 98].

Mole

Le groupe dédié aux systèmes distribués, au sein de l'Institute of Parallel and Distributed High-Performance Systems à l'Université de Stuttgart, dirigé par Kurt Rothermel, était le premier à développer une plate-forme basée sur Java supportant les agents mobiles [Mole]. Récemment, une troisième version de leur système, appelé Mole, a été publiée sur le Web. A travers les versions différentes de la plate-forme, la stabilité et la fonctionnalité du système ont continuellement crû.

Odyssey

Le successeur de Telescript, développé également par General Magic, s'appelle Odyssey [Odyssey]. Différent du premier, ce dernier est basé sur Java. Il est largement inspiré par Telescript, mais n'a pas exactement la même fonctionnalité. Par exemple, Odyssey ne supporte pas le transfert de l'état d'exécution. Le système n'a toujours pas dépassé les versions *bêta*, et ne joue visiblement pas le rôle important que jouait Telescript dans la stratégie de General Magic. Répétons, toutefois, que Danny Lange a joint l'équipe.

Telescript

Telescript, développé par General Magic, peut être considéré comme la référence dans le domaine [White 95] [White 96]. Il est parlant que son inventeur, Jim White, tient un brevet sur la notion même d'agents mobiles [White 97]. Telescript était un des seuls systèmes supportant le transfert de l'état d'exécution des agents. Toutefois, le système, conçu pour une application industrielle, exigeait des ressources importantes, était plutôt cher (5000 USD par moteur), et offrait une fonctionnalité pas encore demandée par le marché. Par conséquent, le système n'existe plus et sa valeur est surtout historique.

Voyager

Voyager est un produit commercial proposé par ObjectSpace [Voyager]. Il s'agit d'un système « branché » incorporant plusieurs normes industrielles populaires à l'heure actuelle, notamment CORBA, DCOM et Distributed JavaBeans. Différent des autres systèmes, il ne s'agit pas d'un système dédié uniquement aux agents mobiles : toute sorte de développement distribué est supporté. Dans leur documentation, cependant, le concept d'agent mobile est explicitement présenté. Puisqu'il s'agit d'un système bien soigné, d'une nature générale, et compatible avec les normes existantes, sa popularité est croissante.

Systèmes moins connus

— Ara, Agents for Remote Action, est un système développé par le Distributed Systems Group du département informatique à l'Université de Kaiserslautern en Allemagne [Ara]. Cette plate-forme supporte l'exécution sécurisée d'agents mobiles à travers des réseaux hétérogènes. Comme dans Telescript, la migration préserve l'état d'exécution des agents. Le but principal du projet est de réaliser la mobilité en exploitant un maximum de modèles et de langages informatiques existants. Notons que l'implémentation de la plate-forme n'est pas encore finalisée.

— JavaNetAgents, un système basé sur Java, est développé par Walter Merlat et Claude Seyrat au laboratoire LIP6 de l'université Pierre et Marie Curie (Paris VI) [Merlat 97a] [Merlat 97b] [Merlat 98]. Il s'agit d'un système expérimental, toutefois doté de fortes fonctionnalités de sécurité, qui a été utilisé dans l'étude de l'adaptation dynamique de l'organisation dans les Systèmes Multi-Agents. Notons que ce sujet de recherche est proche du nôtre. Toutefois, la solution proposée est très différente, basée sur l'utilisation d'agents spécialisés dans l'organisation d'autres agents.

— Java-To-Go, un des premiers systèmes publiés sur le Web, est une création de Weiyi Li et David Messerschmitt du Department of Electrical Engineering and Computer Sciences à l'University of California at Berkeley [Java-To-Go]. Comme le suggère le nom, le système est basé sur Java. Il est d'une nature plus expérimentale que la plupart des systèmes présentés ici.

— MAP, Mobile Assistant Programming, est un système développé par Stéphane Perret et Andrzej Duda du laboratoire LSR de l'IMAG [MAP] [Perret 96a] [Perret 96b] [Perret 96c]. Un des intérêts principaux de ce système, utilisant le langage Scheme, est le fait qu'il incorpore la notion de stations (sites) nomades.

— TACOMA, Tromsø And COrnell Moving Agents, est le résultat d'une collaboration entre les départements informatiques de l'Université de Tromsø, Norvège, Cornell University, Ithaca, New York et l'University of California at San Diego [TACOMA]. La version 1.2 de la plate-forme est basée sur UNIX et TCP et supporte des agents écrits en C, Tcl/Tk, Perl, Python et Scheme. Plusieurs applications sont en phase de construction.

Une courte évaluation

Ci-dessus, nous avons présenté un aperçu des plates-formes principales. Toutefois, il en existe d'autres et un certain nombre d'alternatives sont en phase de développement. Nous pouvons nous attendre à l'introduction continue de nouveaux systèmes dans les années qui viennent.

Pour nous, trois critères de sélection règnent : la distribution, la stabilité et la sécurité. Tenant compte de ces critères, les Aglets d'IBM constituent un des meilleurs choix. Toutefois, nous pouvons nous attendre à une forte concurrence de systèmes plus récents exploitant les nouvelles possibilités de Java et les normes de plus en plus établies de l'informatique distribuée. A court terme, le système Voyager d'ObjectSpace offre une alternative aux Aglets, même si le premier n'est pas autant orienté vers les agents et les Systèmes Multi-Agents que le deuxième.

Pour l'implémentation de FRIENDS Online MAI, décrit dans le chapitre IV.2, nous avons retenu les Aglets. Sa distribution, son ancienneté et son orientation vers les agents nous ont motivé dans le choix de ce système.

Les normes et les agents mobiles

Soulignons encore une fois que l'utilité d'une plate-forme supportant les agents mobiles est proportionnelle à sa distribution. Par conséquent, la normalisation des plates-formes, d'une manière institutionnelle ou autre, est cruciale. Par contre, une prolifération de normes, tourmentant souvent les systèmes de communications, peut retarder sensiblement l'intégration des agents mobiles.

Deux organismes importants proposent des normes pour les agents mobiles : l'Object Management Group (OMG) [OMG] et la Foundation for Intelligent Physical Agents (FIPA) [FIPA].

La première norme, originalement appelé « Mobile Agent Facility » (MAF), et récemment rebaptisé « Mobile Agent System Interoperability Facilities » (MASIF), a été soumis par les entreprises Crystaliz, General Magic, GMD FOKUS, et IBM, avec le support de l'Open Group [Chang 97] [Milojicic 98]. La norme est très liée à CORBA et est centrée sur l'interopérabilité entre des plates-formes différentes supportant la mobilité. Elle traite de la gestion des agents par des administrateurs, du transfert des agents, des noms des agents et des systèmes et de la syntaxe des types et des locations des agents.

Proposant la deuxième norme, la FIPA est une organisation relativement récente à laquelle contribuent, à un niveau stratégique, un grand nombre d'entreprises. Le brouillon FIPA98 0.2, *Part* 11, appelé « Agent Management Support for Mobility » propose une norme pour la mobilité, très liée aux autres spécifications de la FIPA.

Au séminaire international « Mobile Agents '98 », Danny Lange, un des experts le plus respecté dans le domaine, a exprimé un certain scepticisme à propos des deux normes proposés [Lange]. En ce qui concerne la norme de l'OMG, il argumente que nous avons besoin d'une API (*Application Programmers' Interface*) normalisée, plutôt que d'une norme d'interopérabilité. Quand à la norme de la FIPA, il la trouve prématurée et encore très universitaire.

Danny Lange prend une vue très pragmatique de la situation. Apparemment, ce qui compte pour lui, c'est que ça marche. Nous partageons, à un certain degré, ce point de vue. Il est difficile de normaliser ce qui n'existe pas encore. Pour développer les agents mobiles et pour vraiment comprendre les problèmes impliqués, il nous faut des vrais systèmes utilisés à une échelle importante. Une fois de tels systèmes existent, nous pouvons essayer d'établir des normes plus sophistiquées.

Notons que ce processus de normalisation ne sera peut-être pas effectué par un organisme de standardisation officiel. En effet, les agents mobiles seront surtout utilisés sur l'Internet et les systèmes sur ce réseau sont

notoirement difficiles à normaliser d'une manière institutionnalisée. Prenons, par exemple, la standardisation de HTML. En réalité, cette standardisation s'effectue *ad hoc*, et elle est guidée par les activités plus ou moins indépendantes de Microsoft et de Netscape. Les organisations responsables pour la standardisation du Web, le World Wide Web Consortium et l'Internet Engineering Task Force, souvent ne peuvent faire que formaliser les normes *ad hoc* existants [Berghel 98]. Puis, à cause des mécanismes du marché, les fabriquants ne suivent pas nécessairement ce qui est proposé par ces organisations [Zelnick 98]. Toutefois, il est très probable que - un jour - la technologie de la mobilité se normalisera, puisque tout le monde en profitera. Par contre, il est dur de prédire quand et comment cette normalisation s'effectuera.

III.3.4 La technologie des composants

Le modèle des composants informatiques propose de construire des programmes en assemblant un nombre d'éléments informatiques, appelés composants, qui sont *à la fois*, testés, indépendants et remplaçables. Le processus d'assemblage est comparable à celui effectué par un électronicien assemblant une radio à partir de composants discrets. Pendant la dernière décennie, plusieurs implémentations et normes de cette technologie ont été proposées. En 1995, les normes les plus importantes étaient OpenDoc de Component Integration Laboratories, OLE2 et OLE de Microsoft, et CORBA de l'Object Management Group (OMG) [Adler 95].

Depuis, la technologie des composants a gagné énormément d'importance. Toutes les plates-formes de programmation (*Integrated Development Environment* ou IDE) récentes sont fortement orientées vers la programmation par composant. Au niveau de la technologie et des normes, les choses ont également changé : OpenDoc et Component Integration Laboratories n'existent plus, OLE2 et OLE ont évolué vers ActiveX/DCOM, le succès de Java a produit JavaBeans - une technologie de composants très populaire - et l'OMG a modifié la norme CORBA afin d'intégrer ces nouveaux développements guidés par le marché.

Des différences importantes existent entre les technologies et les normes différentes. Par exemple, ActiveX/DCOM est indépendant du langage utilisé, mais dépendant de la plate-forme [Chappel 96]. Par contre, JavaBeans est indépendant de la plate-forme, mais ne supporte que Java [JavaBeans]. Encore différente, CORBA est une norme se situant à un niveau supérieur, offrant des spécifications plutôt qu'une technologie concrète [OMG]. En tout cas, une certaine convergence peut être observée qui sera bénéfique pour les utilisateurs des composants.

A l'heure actuelle, l'assemblage de composants en des systèmes complets est réalisé par les développeurs et les utilisateurs. Toutefois, dans certains cas, l'auto-organisation des composants, même à travers un réseau, est à la fois désirable et techniquement réalisable. Par exemple, un correcteur d'orthographe, continuellement personnalisé par les membres d'une équipe pourrait migrer d'éditeur à éditeur (et même se cloner) selon les besoins des utilisateurs. Dans ce cas, le composant est devenu relativement autonome et peut être vu comme un agent.

Considérons la notion d'autonomie dans un sens large, pas trop différente des notions d'indépendance ou même d'identité. Dans ce sens, un composant est une version autonome d'un objet, qui peut être vu comme une version autonome d'une structure de données, qui peut être considérée comme une version autonome d'un ensemble aléatoire de données. Puis, si nous extrapolons l'évolution historique « variables » → « composants », la série complète devient « variables » → « structures » → « objets » → « composants » → « agents ».

Nous croyons que les composants préparent la route vers les agents et les Systèmes Multi-Agents. Il est vraisemblable que, dans un avenir proche, la fonctionnalité offerte par des systèmes comme Voyager sera intégrée dans tous les systèmes de gestion importants et que des composants autonomes pourront, dans les limites de sécurité imposées, migrer librement de site en site. A ce moment, les agents mobiles feront partie de notre vie quotidienne.

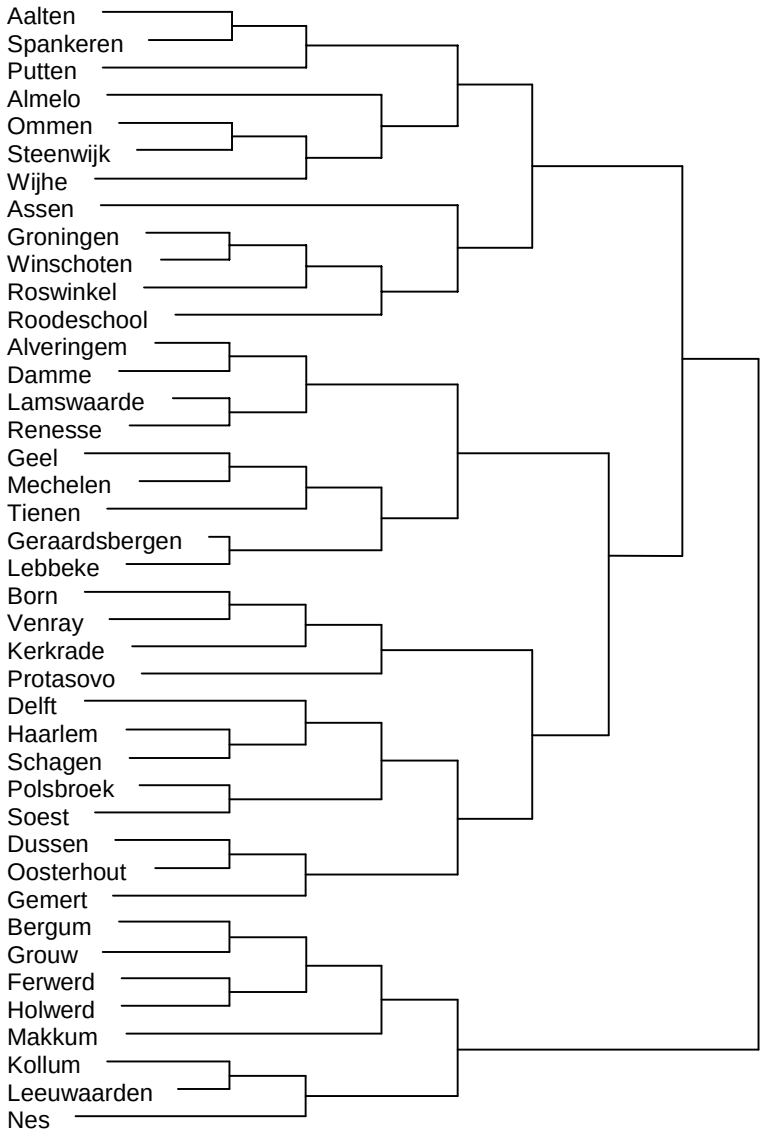


Figure 1 : Exemple de regroupement hiérarchique.

L'algorithme agglomératif de base, très simple, peut être formulé comme suit [Duda 73] :

1) $n = n_{points}$ et $G_i = \{x_i\}$, $i = 1, \ldots, n_{points}$. n_{points} est le nombre de points de données, n est le nombre de groupes. G_i représente le groupe i, x_i représente le point de données (le vecteur) i.
2) Tant que $n > n_{désiré}$, répéter :
3) Trouver (G_i, G_j), le couple de groupes différents dont $d(G_i, G_j)$ est minimal.
4) Fusionner (G_i, G_j) dans le groupe G_i, écraser G_j, et soustraire 1 de n.

Dans son implémentation simple, l'algorithme ci-dessus est de complexité $O(n^3)$. En effet, la recherche du meilleur couple (étape 3) est typiquement de complexité $O(n^2)$, et elle se trouve dans une boucle de complexité $O(n)$.

Le comportement de cet algorithme est directement dépendant du choix de la mesure de distance entre deux groupes. Couramment, une des mesures suivantes est utilisée :

$$d_{\min}(G_i, G_j) = \min_{x \in G_i, y \in G_j} d(x, y) \quad \text{(minimum)}$$

$$d_{\max}(G_i, G_j) = \max_{x \in G_i, y \in G_j} d(x, y) \quad \text{(maximum)}$$

$$d_{moyenne}(G_i, G_j) = \frac{1}{n_i n_j} \sum_{x \in G_i} \sum_{y \in G_j} d(x, y) \quad \text{(moyenne)}$$

$$d_{centroïde}(G_i, G_j) = d(\frac{1}{n_i} \sum_{x \in G_i} x, \frac{1}{n_j} \sum_{y \in G_j} y) \quad \text{(centroïde)}$$

Il existe d'autres mesures, plus complexes, telle que la mesure de Ward, également appelée la mesure de « variance minimale » ou « somme de carrés ». Pour toutes ces mesures, Lance et Williams ont développé un algorithme de complexité $O(n^2)$ effectuant le regroupement hiérarchique agglomératif d'un ensemble de données [Lance 67]. Soulignons que cet algorithme ne marche pas pour tout mesure de distance. En particulier, la mesure doit être une *métrique* qui peut être exprimé dans la forme suivante :

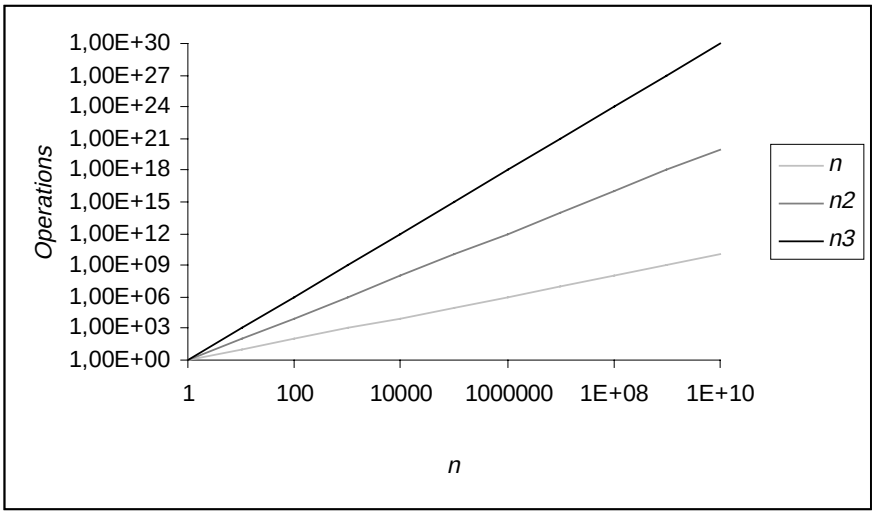
$$d((G_i, G_j), G_k) = \alpha_i d(G_i, G_k) + \alpha_j d(G_j, G_k) + \beta\, d(G_i, G_j) + \gamma \left| d(G_i, G_k) - d(G_j, G_k) \right|$$

La distance entre un nouveau groupe formé - selon le critère utilisé - et un troisième doit être exprimable comme une combinaison linéaire des distances entre les deux sous-groupes et le troisième, dont les coefficients sont en fonction de la taille des groupes. En particulier, pour les mesures présentées ci-dessus, les coefficients sont les suivants :

Mesure	α_i	α_j	β	γ
minimum	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
maximum	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
moyenne	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	0	0
centroïde	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	$-\frac{n_i n_j}{(n_i + n_j)^2}$	0
Ward	$\frac{n_i + n_k}{n_i + n_j + n_k}$	$\frac{n_j + n_k}{n_i + n_j + n_k}$	$-\frac{n_k}{n_i + n_j + n_k}$	0

Tableau 1 : Les coefficients de Lance et Williams.

Comme l'illustre le graphique 2, montrant l'évolution de n, n^2 et n^3 en fonction de n, la complexité $O(n^2)$ des algorithmes optimisés limite l'application du regroupement hiérarchique à des ensembles de données de taille relativement modeste. Ceci a motivé les chercheurs du domaine pour développer des méthodes approximatives, utilisant - par exemple - des algorithmes génétiques [Fränti 97], ou pour concevoir des méthodes adaptées à des architectures parallèles [Olson 94].



Graphique 2 : Complexité des algorithmes.

IV.1.2 Les SMAM augmentés et le regroupement hiérarchique

Il existe une forte correspondance entre l'évolution naturelle d'un SMAM augmenté, dont les agents sont guidés par le comportement « qui se ressemble, s'assemble », et le regroupement hiérarchique. Dans le SMAM, les attributs des agents correspondent aux points de données.

En particulier, le comportement de FRIENDS, optimisant son harmonie et - si possible - son entropie peut être interprétée comme un processus de regroupement. Toutefois, il existe des différences importantes entre ce système et un système de regroupement classique.

Primo, FRIENDS est conçu comme un système dynamique, accueillant continuellement des nouveaux agents et permettant, à tout moment, des changements des attributs des agents. D'un point de vue conceptuel, cette fonctionnalité est d'une grande importance, car elle est à la base de la boucle fermée entre le système et les utilisateurs, leur permettant d'établir un vocabulaire commun. Nous reprendrons cette problématique dans le prochain chapitre, lorsque nous présenterons FRIENDS Online. Le regroupement classique, par contre, présuppose que l'ensemble de données est statique. L'efficacité relative des algorithmes de complexité $O(n^2)$ est, entre autre, dépendante de la nature statique de la matrice des distances entre les points de données.

Secundo, dans FRIENDS, le comportement des agents est basé sur une fonction de ressemblance, plutôt que sur une fonction de distance. Cette fonction de ressemblance correspond, dans le contexte du regroupement, à une mesure de similarité. Toutefois, les algorithmes classiques sont basés sur une mesure de distance, ayant les propriétés d'une métrique. Est-ce que nous pouvons exprimer notre mesure de ressemblance comme une mesure de distance ?

La mesure de distance de FRIENDS

Nous avons défini la ressemblance entre deux agents comme le nombre de mots-clés communs entre eux. Au niveau conceptuel, dans FRIENDS, l'espace des mots-clés n'est pas contraint : les utilisateurs peuvent introduire n'importe quel mot-clé. Par conséquent, l'espace des données est, au niveau conceptuel, d'une dimensionnalité infinie. Donc, la mesure de distance « Manhattan », par exemple, doit être exprimée comme suit :

$$d(x, y) = \sum_{i=0}^{\infty} |x_i - y_i|$$

Notons que, dans le cas général de FRIENDS, les éléments des vecteurs prennent des valeurs binaires, '1' indiquant la présence du mot-clé, 'X' indiquant son absence. La notion de « anti-intérêt », comme elle est utilisée dans FRIENDS Offline, ne fait pas partie du modèle général de FRIENDS. Donc, dans ce chapitre, nous utilisons le symbole '0', plutôt que le symbole 'X' pour présenter l'absence d'un mot-clé.

La nature infinie du nombre de dimensions de l'espace de données peut, d'un point de vue mathématique et algorithmique, poser certains problèmes, par exemple liés à la normalisation des données. Par contre, dans un système réel, opérationnel pendant une durée finie, le nombre de dimensions est limité par les ressources disponibles et il est donc toujours fini.

Reste toujours la question de savoir si notre mesure de ressemblance correspond à une mesure de distance. Etudions les exemples suivants :

A	***B***	***R*(*A*, *B*)**	***d*(*A*, *B*)**
{a}	{p}	0	2
{a, b, c}	{p, q, r}	0	6
{a}	{a}	1	0
{a, b, c}	{a, b, c}	3	0
{a, b, c, d, e, f}	{a, b, c, p, q, r}	3	6

Tableau 2 : Comparaison entre la ressemblance et la distance « Manhattan ».

Dans le tableau 2, *A* et *B* représentent deux agents dont les listes de mots-clés sont affichées. *R*(*A*, *B*) et *d*(*A*, *B*) représentent respectivement la ressemblance et la distance « Manhattan » entre les deux agents (points de données). Le tableau montre qu'il n'existe pas de relation évidente entre les deux mesures. Par exemple, pour une valeur fixe de ressemblance (3), la distance « Manhattan » peut prendre une valeur inférieure (0) ou une valeur supérieure (6). A la base de ce problème se trouve la différence sémantique entre le symbole '0' et le symbole '1'. La mesure de distance « Manhattan » ne tient pas en compte cette différence.

Dans la section précédente, nous avons présenté une formule permettant la transformation d'une mesure de similarité en une mesure de distance. Est-ce que nous pouvons l'appliquer à *R*(*A, B*) ?

Il se pose le problème que $R(A, B)$ doit être normalisé entre 0 et 1. Autrement dit, nous devons formuler une mesure normalisée $R_{normalisé}(A, B) = R(A, B) / N$ où N représente le nombre de dimensions. Cette mesure est inutile dans le cas général de FRIENDS, où le nombre de dimensions est infini et la mesure, presque toujours, 0.

Limitons-nous au cas caractérisés par une dimensionnalité finie. Le tableau suivant montre quelques exemples pour N = 26.

A	***B***	***R(A, B)***	$\frac{R(A,B)}{N}$	$\frac{N}{R(A,B)}-1$
{a}	{p}	0	0	∞
{a, b, c}	{p, q, r}	0	0	∞
{a}	{a}	1	0.0385	25
{a, b, c}	{a, b, c}	3	0.1154	7.6667
{a, b, c, d, e, f}	{a, b, c, p, q, r}	3	0.1154	7.6667

Tableau 3 : Mesure de distance déduite de la ressemblance.

Reste à savoir si cette mesure de distance est conforme aux propriétés d'une métrique. En particulier, est-ce que $d(A, C) \leq d(A, B) + d(B, C)$? Imaginons que - pour n'importe quel N - A ai un seul mot-clé en commun avec B, B un seul mot-clé en commun avec C, et A aucun mot-clé en commun avec C (le tableau 4 montre un exemple). En conséquence, $d(A, B)$ et $d(B, C)$ sont fini et $d(A, C)$ est infini. Dans ce cas, l'inégalité souhaitée n'est pas respectée. Donc, la mesure de distance que nous avons définie n'est pas une métrique.

A	***B***	***C***	$\frac{N}{R(A,B)}-1$	$\frac{N}{R(B,C)}-1$	$\frac{N}{R(A,C)}-1$
{a, b, c}	{c, d, e}	{e, f, g}	25	25	∞

Tableau 4 : $d(A, C) > d(A, B) + d(B, C)$.

Dans le contexte du regroupement automatique de données, nous ne devons pas seulement définir une mesure de distance entre les points de données, mais également une mesure de distance entre les groupes. Dans le cas de FRIENDS, la ressemblance entre deux agents composés est calculée de la même manière que la ressemblance entre deux agents atomiques : en comptant le nombre de mots-clés en commun, sachant que la liste de mots-clés d'un agent composé est égale à l'intersection des listes de mots-clés des deux agents composants. Donc, à ce niveau également, nous n'avons pas à notre disposition une mesure évidente de distance ayant les propriétés d'une métrique.

Un problème NP-difficile

Nous avons montré que les systèmes de regroupement hiérarchique, et le système « FRIENDS » se ressemblent énormément, mais diffèrent sur quelques points importants. FRIENDS est un système dynamique et le nombre de dimensions de l'espace des attributs n'est pas constant. De plus, FRIENDS est basé sur une mesure de similarité difficile à traduire en une mesure de distance ayant les propriétés d'une métrique.

Une des implications principales de ces différences est le fait que les algorithmes optimisés du regroupement hiérarchique ne peuvent pas directement être appliqués à FRIENDS. Dans FRIENDS Numbercruncher et dans FRIENDS Online, nous utilisons un algorithme de complexité $O(n^3)$, plutôt que $O(n^2)$.

Cet algorithme donne des résultats satisfaisants, mais pas nécessairement optimaux. En effet, comme l'ont montré Frédéric Maffray et Myriam Preissmann, le problème de l'organisation optimale d'un SMAM augmenté de type FRIENDS est *NP-difficile* (voir annexe). Donc, dans un premier temps, nous avons cherché des solutions satisfaisantes plutôt qu'optimales.

A un niveau pratique, même une complexité d'$O(n^3)$, voire $O(n^2)$, s'avère très lourde. En effet, les algorithmes optimisés utilisés dans le regroupement hiérarchique restent coûteux et un grand nombre de chercheurs sont impliqués dans la recherche d'alternatives. Nous croyons que notre modèle peut être utile dans cette recherche. Il introduit le point de vue « Agent » et « Système Multi-Agents » dans la problématique, suggérant une nouvelle approche à l'implémentation distribuée de systèmes de regroupement. Dans le chapitre II.3, nous avons proposé l'implémentation des SMAM purs comme un *benchmark* pour des agents cognitifs. Il s'agit d'un problème très proche de l'implémentation des SMAM augmentés du type FRIENDS et une meilleure compréhension du premier mènera sans doute à une meilleure compréhension du deuxième. Il nous semble que le modèle des SMAM facilitera l'application naturelle de techniques de l'Intelligence Artificielle dans le domaine du regroupement hiérarchique.

Toutefois, notons que le regroupement effectué dans FRIENDS est d'une nature relativement exotique, impliquant un très grand nombre de dimensions semi-binaires ('1' et 'X', plutôt que '1' et '0') et une grande dynamique du système. Nous nous attendons à une importance croissante de ce type de systèmes, surtout dans le contexte « Internet » et « Intranet ». Pourtant, le modèle peut également servir dans un contexte statique. Dans la section suivante, nous présenterons FRIENDS Numbercruncher, une application de FRIENDS dédiée au regroupement hiérarchique de données statiques.

IV.1.3 FRIENDS Numbercruncher

Le logiciel « FRIENDS Numbercruncher », version 1.0, permet d'effectuer un regroupement hiérarchique de données statiques, selon les principes des SMAM augmentés du type « FRIENDS ». Autrement dit, FRIENDS Numbercruncher est une version de FRIENDS, mais limitée à des populations statiques. Dans cette section, nous présentons l'essentiel du logiciel. Son manuel complet constitue la deuxième annexe de cette thèse.

Comme FRIENDS Offline 1.7, FRIENDS Numbercruncher 1.0 a été développé avec Microsoft Visual C++ 5.0. Nous avons argumenté notre choix de cet environnement dans le chapitre III.2. Le choix d'un système générant du code efficace était crucial, car FRIENDS Numbercruncher est conçu pour organiser des populations

relativement grandes. Le nom « FRIENDS *Numbercruncher* » a été choisi pour illustrer la nature computationnellement intensive de l'application.

Le format des fichiers contenant les données d'entré a été spécifié par le CNET, le Centre de Recherche et Développement de France Télécom. Ceci a permis l'utilisation du logiciel avec des données compilées au sein de France Télécom.

La figure 2 montre la fenêtre principale de l'application. Son utilisation se décompose en 5 phases, activées par les 5 boutons, étiquetés « Start Analysis », « Build Dictionary », « Create Population », « Start Organisation » et « Generate Report ». Détaillons ces étapes différentes.

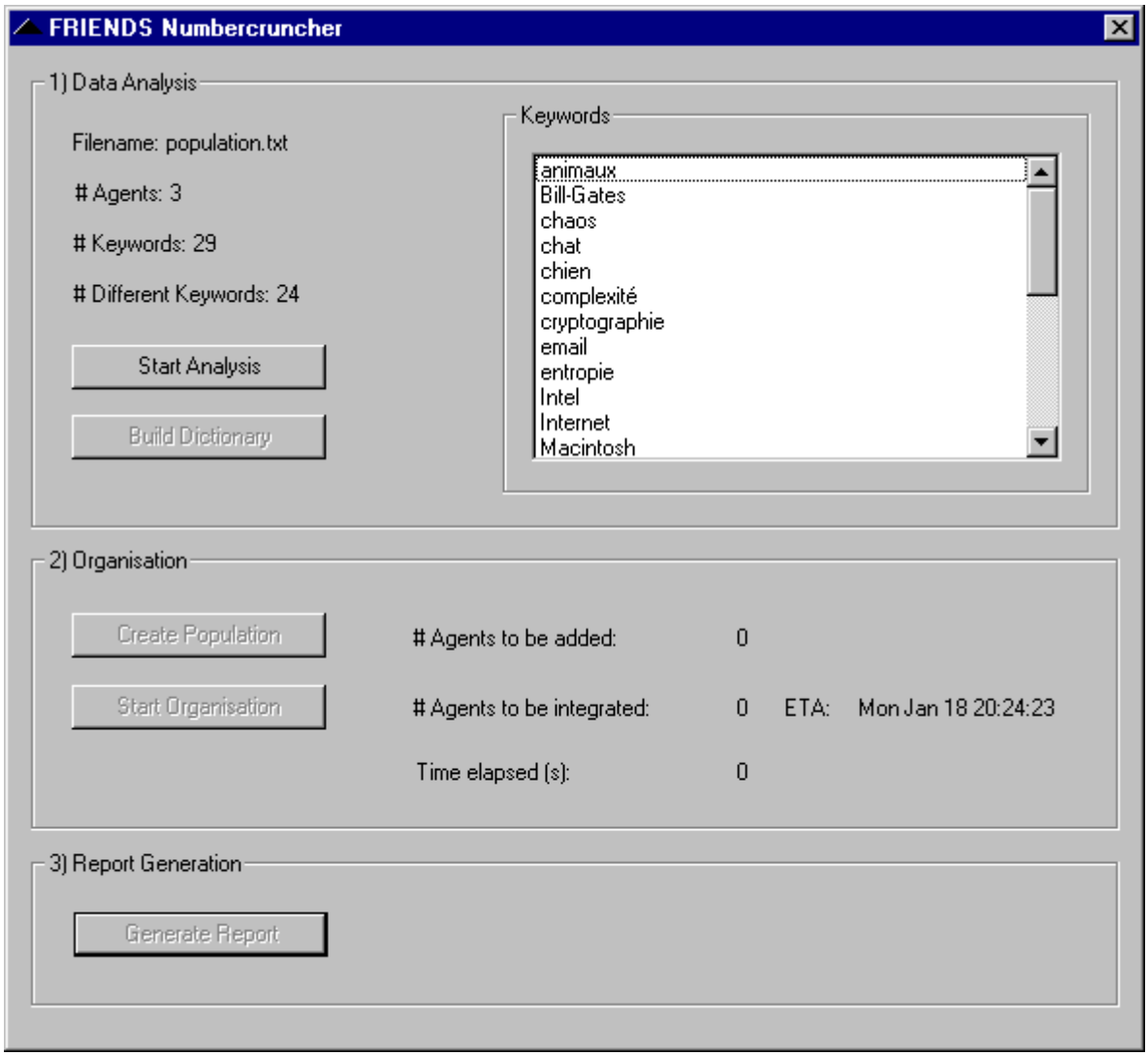


Figure 2 : Fenêtre principale de FRIENDS Numbercruncher.

Première phase : « Start Analysis »

Dans cette première phase, le programme invite l'utilisateur à charger un fichier de texte contenant la description de la population. Le format de ce type de fichier est conforme au format spécifié par le CNET : les mots-clés sont séparés par des espaces ou des fins de lignes. Les listes différentes, correspondant aux agents différents,

sont séparées par des lignes vides. Illustrons ces conventions en présentant le contenu d'un fichier (d'une taille très modeste) conforme au format :

```
chat chien plage vacances voyages soleil
poissons vacances animaux

ordinateurs Internet Web
Bill-Gates email photographie PC Macintosh Intel
cryptographie complexité chaos

systèmes-dynamiques chaos Vie-Artificielle entropie
photographie poissons Seychelles voyages
```

Le fichier contient les listes de mots-clés de trois utilisateurs. Si un « mot-clé » est composé de plusieurs mots, les composants sont liés par un tiret. Notons que les listes peuvent contenir des doubles.

La population initiale n'est pas structurée et ne correspond pas à un SMAM augmenté. Par contre, le résultat de son organisation en est effectivement un.

Deuxième phase : « Build Dictionary »

Dans la deuxième étape, un dictionnaire global est construit, liant chaque mot-clé à un nombre entier. Si deux mots-clés sont lexicalement identiques, ils sont liés au même nombre, éliminant les doubles. L'utilisation d'un dictionnaire accélère sensiblement l'organisation de la population. Notons que, si le nombre de mots-clés est grand, la construction du dictionnaire peut prendre un certain temps.

Troisième phase : « Create Population »

Dans la troisième phase, une représentation optimale - en considérant le temps de calcul - de la population est créée. La liste de mots-clés de chaque agent est convertie, en utilisant le dictionnaire global, en une liste de nombres entiers tous différents. Si le nombre total de mots-clés est grand, ce processus aussi peut prendre un certain temps.

Quatrième phase : « Start Organisation »

Une fois la population créée, elle peut être organisée. L'algorithme utilisé dans la version 1.0 de FRIENDS Numbercruncher est le suivant :

```
1) Attribuer à chaque membre de la population un agent atomique.
2) Tant que la population compte plus qu'un seul SMAM, répéter :
3)       Trouver, dans la population, les deux SMAM les plus
         ressemblants.
4)       Fusionner ces deux SMAM en créant un agent composé
         représentant leur couple.
5)       Remplacer, dans la population, les deux SMAM originaux par
         le nouvel agent composé.
```

Notons que l'algorithme utilisé correspond directement à l'algorithme de base du regroupement hiérarchique, et qu'il est également de complexité $O(n^3)$. Comme l'illustre l'exemple des figures 3 et 4, l'algorithme ne trouve pas nécessairement la solution optimale. La figure 3 montre une population organisée selon notre algorithme. La

figure 4 montre son organisation optimale, qui est différente. Puisque notre algorithme génère des résultats satisfaisants et puisque l'optimisation de l'organisation est un problème NP-difficile, son utilisation nous semble raisonnable.

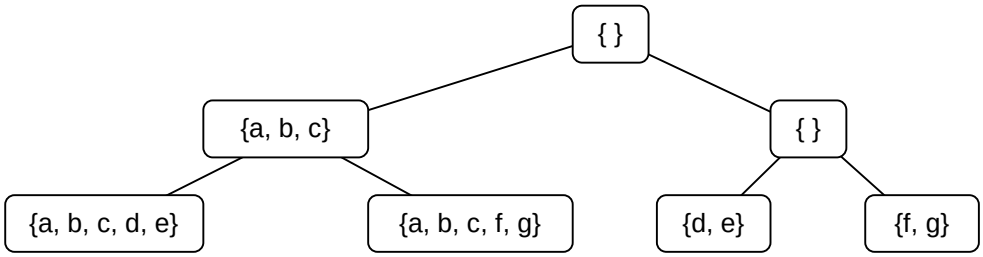


Figure 3 : Organisation sous-optimale d'une population.

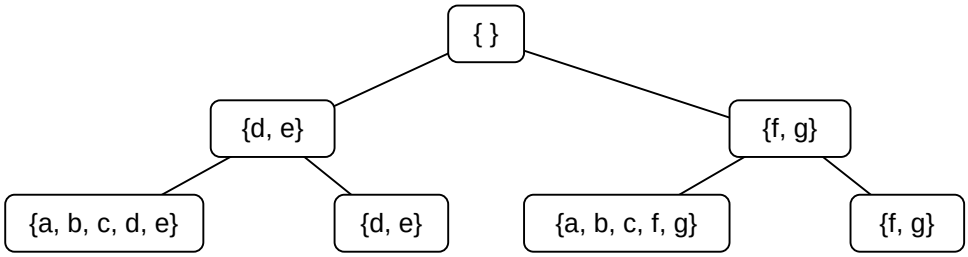


Figure 4 : Organisation optimale de la population de la figure 3.

Cinquième phase : « Generate Report »

L'étape finale consiste en la génération d'un rapport. Dans ce rapport, le dendrogramme correspondant à la structure du SMAM final est représenté d'une manière alphanumérique. Chaque ligne représente un agent. A chaque ligne, la liste de mots-clés est précédée par un nombre de tirets, correspondant au niveau de l'agent. La présence d'une astérix avant la liste de mots-clés indique que l'agent est atomique

Illustrons cette présentation à l'aide d'un exemple :

```
<ligne vide>
-* Bill-Gates chaos complexité cryptographie email Intel Internet
   Macintosh ordinateurs PC photographie Web
-  poissons voyages
--* animaux chat chien plage poissons soleil vacances voyages
--* chaos entropie photographie poissons Seychelles systèmes-
    dynamiques Vie-Artificielle voyages
```

Notons que, dans cet exemple, certaines lignes dépassent la largeur de la page et ont du être tronquées. En général, d'un point de vue ergonomique, le format de la présentation n'est pas idéal. Dans un premier temps, nous nous sommes concentrés sur la fonctionnalité globale de l'application et, en ce qui concerne la présentation des rapports, des améliorations peuvent être effectuées.

Performance

Nous avons testé FRIENDS Numbercruncher en utilisant un nombre de populations différentes. Ces essais nous ont assuré que le comportement du programme est correct.

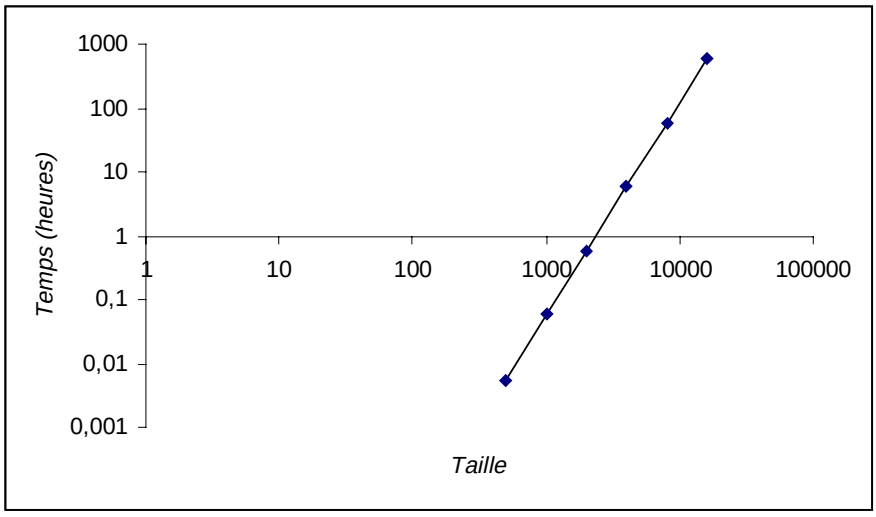
Pour un nombre donné de mots-clés par agent (en moyenne), le temps d'exécution de l'organisation est en fonction du nombre d'agents, et la complexité $O(n^3)$ de l'algorithme est bien reconnaissable. Sur un Dell Dimension XPS D300 (à base d'un Pentium II à 300 MHz), et sous Windows NT 4.0, des résultats typiques sont, pour des populations d'agents ayant, en moyenne, 10 mots-clés :

Taille de la population	**Temps d'organisation**
500 agents	20 secondes
1000 agents	219 secondes
2000 agents	2075 secondes

Tableau 5 : Temps d'organisation.

Lorsqu'on double le nombre d'agents, le temps est augmenté par un facteur 10, plutôt que par le facteur 8 attendu. La baisse d'efficacité des *caches* du processeur (en fonction de la taille des données) est sans doute un facteur important contribuant à ce phénomène [Booth 97]. Pour la même raison, le *Estimated Time of Arrival*, affiché par le programme est souvent un peu pessimiste (car l'estimation est basée sur le temps déjà écoulé).

Le graphique 3 montre l'extrapolation des temps présentés ci-dessus. Selon cette extrapolation, la réorganisation d'une population comptant 10000 agents prend déjà plusieurs jours sur la plate-forme utilisée. Lorsqu'on souhaite organiser des populations de cette taille ou de taille supérieure, une approche différente doit être suivie. Plusieurs options se présentent.



Graphique 3 : Extrapolation des temps d'organisation.

Nous pouvons essayer de trouver un algorithme d'une complexité inférieure, préférablement au-dessous de $O(n^2)$. Ceci ne nous semble pas une tâche facile, car les algorithmes connus de regroupement, exploitant les propriétés de la métrique utilisée, sont tous d'une complexité aux alentours de $O(n^2)$.

Alternativement, ou parallèlement, nous pouvons essayer de distribuer le processus d'organisation sur plusieurs machines. Les solutions distribuées, proposées pour les cas classiques du regroupement hiérarchique, ne sont pas directement utilisables dans le contexte de FRIENDS, car elles exigent des mesures de distance bien spécifiques. Par contre, la nature des SMAM augmentés permet de distribuer naturellement les agents et donc l'application. Cette solution exige des algorithmes distribués relativement efficaces, impliquant un certain degré d'intelligence de la part des agents. En effet, le benchmark proposé dans le chapitre II.3 peut être étendu aux SMAM augmentés tels que FRIENDS.

Vu la nature NP-difficile du problème, la réalisation d'organisations optimales est peu réaliste et la recherche de solutions acceptables, bien que sous-optimales, s'impose. Ceci constitue un argument secondaire pour appliquer la programmation orientée « Agent » et « Système Multi-Agent », menant typiquement à des solutions naturelles, mais pas nécessairement optimales.

IV.1.4 Evaluation

Dans la section précédente, nous avons présenté notre évaluation technique de FRIENDS Numbercruncher. A un niveau plus appliqué, une première série d'essais a été effectuée par les chercheurs du CNET. Ces essais sont basés sur des données compilées au sein de France Télécom dans le cadre du service « QuiQuoiOù ».

QuiQuoiOù

QuiQuoiOù est un annuaire des services francophones sur le Web [QuiQuoiOù]. Il est développé par France Télécom comme un service de Wanadoo et il est gratuitement accessible sur le Web. Les utilisateurs peuvent manuellement consulter l'arborescence où formuler des requêtes libres. En octobre 1998, 38000 services étaient catalogués.

Les opérateurs de l'annuaire parcourent systématiquement les nouveaux services francophones et attribuent à chacun une liste de mots-clés. Leur choix des mots-clés est libre. Il classifient également la service dans l'arborescence préexistante de QuiQuoiOù. Le traitement des requêtes libres est basé sur la fusion d'un moteur de recherche documentaire (Topic de Verity [Verity]) et de technologies linguistiques (développées par Erli [Erli]).

Les essais

Dans les essais, FRIENDS Numbercruncher a été utilisé pour construire des classifications basées sur les listes de mots-clés compilées par les opérateurs de QuiQuoiOù. Puis, les classifications résultantes ont été évaluées. Les chercheurs de France Télécom ont effectué trois essais sur respectivement 300, 1258 et 4997 listes de mots-clés décrivant des services.

A cause de sa taille, le troisième test a produit les résultats les plus intéressants. Les données décrivant les 4997 services étaient composées de 146674 mots-clés dont 16384 différents. FRIENDS Numbercruncher a construit le SMAM complet en 70337 secondes (19 heures et demie). La plate-forme est inconnue. Nous avons inclus en

annexe l'arborescence du SMAM jusqu'à niveau 7, identifiant 2^7 = 128 groupes au niveau le plus bas. Les niveaux plus détaillés sont confidentiels et d'une importance secondaire dans cette première évaluation.

A partir du niveau 6, 18 des 64 groupes sont identifiés par un mot-clé, ce qui constitue un résultat très encourageant. De plus, à partir du niveau 7, *chaque* groupe est identifié. Ceci implique qu'un système tel que FRIENDS peut effectivement être utilisé pour l'identification automatique de catégories pertinents.

Le tableau 6 montre la liste des catégories identifiées au niveau 6. Il s'agit en effet de catégories importantes. De plus, elles sont - en théorie - plus utiles que des catégories définies *à priori*, car elles reflètent la constitution de la population (de services).

RELIGION	LINGUISTIQUE
SPORT	POLITIQUE
ALIMENTATION	ZOOLOGIE
LITTERATURE	MUSIQUE
COMMERCE	MEDIA
DROIT	MEDECINE
ART	TRANSPORTS
EDUCATION	TELECOM
INFORMATIQUE	GEOGRAPHIE

Tableau 6 : Catégories identifiées par FRIENDS Numbercruncher.

FRIENDS n'identifie pas seulement des catégories, il les structure également dans une hiérarchie. Nous espérons évaluer cette capacité de structuration ultérieurement. Toutefois, une première inspection des résultats du deuxième essai révèle des éléments intéressants. Par exemple, dans la catégorie GEOGRAPHIE, nous retrouvons des groupes comme CANADA, FRANCE et BELGIQUE, mais également des groupes comme AGRICULTURE et TRANSPORTS. Les sous-groupes semblent être sémantiquement liés aux groupes, mais non nécessairement de la même manière.

Une étude plus approfondie des résultats est nécessaire, mais la première analyse s'avère très encourageante. Les essais de France Télécom ont démontré que le concept de FRIENDS peut - au moins jusqu'à un certain degré - être utile dans un contexte réel.

CHAPITRE IV.2

LA RECHERCHE ET LA CLASSIFICATION D'INFORMATIONS SUR LE WEB

To like and dislike the same things, that is indeed true friendship.
Sallust

IV.2.1 Trouver des informations sur le Web

Comme nous l'avons souligné dans le chapitre I.1, l'explosion de l'Internet constitue un des grands événements de l'histoire de l'informatique. L'Internet peut être étudié selon un grand nombre de points de vue différents, dont deux sont très répandus. D'un coté, il peut être considéré comme une vaste base de *données*. Les systèmes et les techniques présentés dans cette section sont conçus selon cette vision. De l'autre côté, l'Internet peut être vu comme un réseau connectant - par l'intermédiaire des ordinateurs - des millions d'*utilisateurs*. Nous présenterons cette deuxième perspective, soulignant la nature sociale de l'Internet, dans la section suivante.

Dans cette section, nous nous concentrons sur le Web, plutôt que sur l'Internet en général. Le Web et son protocole « HTTP » constituent la partie la plus visible de l'Internet et ils sont de plus en plus utilisés pour implémenter des services traditionnellement offerts par d'autres composants de l'Internet, tels que FTP, IRC ou Gopher.

La plupart des chercheurs en informatique considèrent le Web comme une énorme base de données. Marti Hearst, de l'University of California at Berkeley, nous indique, dans son *column* dans *IEEE Intelligent Systems*, que « Despite the Web current disorganized and anarchic state, many AI researchers believe that it will become the world's largest knowledge base. » [Hearst 98]. Ou encore, Venkat Gudivada et ses collègues nous font savoir, dans *IEEE Internet Computing*, que « The World Wide Web is a very large distributed digital information space. » [Gudivada 97].

Il ne s'agit pas d'une base de données ordinaire. Sa taille, courant été 1998, était estimée entre 400 et 500 millions de documents [Filman 98]. De plus, elle est distribuée sur un grand nombre de machines et pratiquement non organisée. Par conséquent, la consultation de cette base de données pose des problèmes très particuliers. Etudions les solutions principales, développées pour trouver des informations sur le Web. Notons qu'il existe un grand nombre de publications détaillant ces solutions. Par exemple, [Gudivada 97] et le numéro dédié d'*IEEE Internet Computing* [Filman 98] offrent un bon aperçu du domaine.

Les robots

La méthode la plus simple et la plus coûteuse pour trouver des documents sur le Web consiste à lancer un agent statique explorant systématiquement tous les sites du Web, chaque fois qu'un utilisateur formule une requête. Un tel agent est typiquement appelé un **robot** (ou « bot », ou - en anglais - *spider* [Cheong 96]).

Considérant la taille du Web, l'approche décrite ci-dessus n'est pas réaliste. Une approche plus pratique, et très populaire, consiste en la construction d'un index précompilé continuellement mis à jour par des robots. La plupart des services de recherche, typiquement appelés **moteurs de recherche**, sont basés sur ce principe. Ils incluent AltaVista [AltaVista], Excite [Excite], HotBot [HotBot], infoseek [infoseek], Lycos [Lycos] et WebCrawler [WebCrawler]. Il existe plusieurs techniques pour explorer le Web. La méthode la plus populaire découpe le Web selon les noms des domaines et attribue un ou plusieurs robots à chaque partition à explorer.

L'exploration du Web et la communication avec les utilisateurs sont relativement faciles à réaliser. Par contre, la construction de l'index représentant le Web constitue un problème ardu. En effet, afin de pouvoir construire un tel index, nous devons utiliser des méthodes d'indexation nécessitant la description des documents. Cette description est typiquement composée d'éléments objectifs, tels que l'URL ou la date de création, et des éléments non objectifs, directement liés au contenu sémantique. L'extraction des éléments objectifs d'un document ne pose aucun problème. Par contre, il s'avère très difficile de décrire, d'une manière automatique, le contenu d'un document.

Il s'agit du problème de *natural language understanding*, étudié en Intelligence Artificielle, et loin d'être résolu. A cause du problème de l'enracinement des symboles, nous croyons que le problème est complexe et que sa solution complète nécessitera des entités artificielles capables de passer le Test de Turing. Toutefois, dans le domaine du traitement automatique des langues, les chercheurs sont plus optimistes et suggèrent que le problème peut être résolu pour des domaines de discours strictement délimités. Evidemment, dans le contexte de l'Internet, le domaine est extrêmement ouvert et les techniques actuelles de l'Intelligence Artificielle ne sont pas suffisamment puissantes.

En raison de la nature difficile du problème, les méthodes actuelles se fondent sur une analyse *lexicale*, plutôt que sémantique, des documents. En général, ces méthodes sont évaluées en utilisant les critères de *recall* et de *precision*, connus depuis longtemps dans le domaine de la Recherche d'Informations (*Information Retrieval*). Pour une collection de documents donnée et une requête spécifique, ***recall*** est le ratio entre le nombre de documents pertinents trouvés et le nombre total de documents pertinents dans la collection. ***Precision***, par contre, est le ratio entre le nombre de documents pertinents trouvés et le nombre total de documents trouvés. Dans le cas idéal, les deux critères sont maximisés. En pratique, ce but est difficile à atteindre car, dans des systèmes concrets, les deux critères sont souvent contradictoires.

En général, les méthodes utilisées sont de nature statistique. Souvent, elles utilisent la mesure *term-frequency x inverse-document-frequency* pour ordonner les documents d'une collection, selon leur pertinence vis-à-vis d'un nombre de mots (*terms*) donnés. La *term frequency* tf_{ij} est égale au nombre d'occurrences du *term* T_j dans le document D_i. L'*inverse document frequency* est donnée par $\log(N/df_j)$ où df_j est la *document frequency*, étant le nombre d'occurrences du *term* T_j dans la collection de N documents. Donc, la mesure ***term-frequency x inverse-document-frequency*** est donnée par : $w_{ij} = tf_{ij} \log(N/df_j)$. Pour une population et un *term* T_j donnés, plus w_{ij} est important, plus le document D_i est pertinent.

Comme l'illustre la performance relativement faible des moteurs de recherche utilisant l'indexation lexicale, ces méthodes ne peuvent pas remplacer l'indexation sémantique. Ceci a motivé la communauté à développer des mécanismes permettant - au sein des documents - la spécification explicite de leur contenu. En général, ces mécanismes sont basés sur l'utilisation de **meta-données** dans les documents.

Par exemple, dans HTML, des *meta-tags* permettent aux créateurs et aux utilisateurs d'un document de spécifier, dans celui-ci, des couples (*attribut*, *valeur*) détaillant son contenu. L'exemple suivant, extrait de la page à l'URL « http://www.xml.com/xml/pub » (20 janvier 1999), illustre son utilisation :

```
<title>XML.COM - Namespaces in XML</title>
<head>
<META NAME="robots" CONTENT="index,follow">
<META NAME="Description" CONTENT="The "Namespaces in XML"
specification has been formally adopted  by the W3C as a
recommendation. XML.com's Mark Walter explains  why this was needed
and what it will do to increase the adoption of XML.">
<META NAME="Keywords" CONTENT="XML,XML.com,XML,XML Development,XML
Developer,Extensible Markup Language,XSL,XLL,style,stylesheets,style
sheets,css,dsssl,xml tutorials,xml tools,xml parsers,xsl
tutorials,xsl tools,xsl parsers,css resources,style sheet
resources,xml resources,xml rpc,metadata">
<LINK HREF="/xmlstyle.css" type="text/css" rel="stylesheet">
</head>
```

L'*Extensible Markup Language* (XML) est un langage relativement récent, se situant entre HTML et SGML, permettant la description de documents structurés et facilitant l'intégration de meta-données [XML]. Dans le même esprit, le World Wide Web Consortium (W3C) nous propose RDF, le *Resource Description Framework*, une norme pour Web *meta-données* [Lassila 98].

Notons que l'introduction de meta-données dans des documents ne facilite pas seulement leur indexation, mais permet également l'indexation de documents non textuels. Dans un monde de plus en plus multimédia, ceci constitue un avantage important.

Il est difficile de prédire l'évolution des normes de meta-données, *ad hoc* et officielles, mais elles seront toutes limitées par le même problème, à savoir le développement d'ontologies partagées. En effet, l'utilisation de *meta-données* n'est utile que lorsque toutes les parties impliquées attribuent une sémantique similaire aux attributs et aux valeurs utilisés.

La classification semi-automatique

Au lieu d'utiliser des méthodes complètement automatiques pour indexer les documents du Web, on peut tenter de les classifier, jusqu'à un certain degré, à la main. Cette approche est suivie - entre autres - par Yahoo [Yahoo] et Magellan [Magellan], qui nous proposent des ***directories*** rédigés à la main.

Cette méthode est intéressante car elle permet la classification basée sur le contenu réel des documents (ou des sites), plutôt que sur une correspondance de mots-clés. Par contre, la taille énorme du Web implique que, avec des moyens réalistes, seulement une petite partie peut être classifiée. Autrement dit, la *precision* de ces systèmes est importante, mais le *recall* n'est pas très élevé.

Hormis ce problème, l'approche est limitée par le fait que la classification de documents dans une structure arborescente nécessite des compromis. En effet, souvent un document peut être classifié dans plusieurs classes distinctes. Dans ce cas, le classificateur doit choisir *soit* une seule catégorie, *soit* plusieurs et lier les instances différentes du document avec des références (*links*). Aucune solution n'est parfaitement satisfaisante, car la première implique une perte d'informations et la deuxième dilue la classification. De plus, les choix effectués

par les classificateurs sont individuels et ne correspondent pas forcement aux choix qu'effectueraient les utilisateurs.

Les méta-moteurs

Un troisième type de systèmes de recherche pour le Web dirigent les requêtes des utilisateurs vers plusieurs moteurs classiques et retournent une compilation des différents résultats. MetaCrawler [MetaCrawler] et MetaFind [MetaFind] sont des exemples de ces systèmes, typiquement appelés **meta-moteurs**. Ces systèmes peuvent être très utiles, mais n'offrent pas une solution réellement alternative.

IV.2.2 *Community Ware*

Les systèmes traditionnels permettant la recherche d'informations sur le Web sont conçus en se fondant sur la perception que le Web est une vaste base de données, et ils exploitent des techniques utilisées dans le domaine classique de la Recherche d'Informations. Comme nous l'avons indiqué dans la section précédente, cette approche a des limites importantes qui ne sont pas faciles à franchir. Toutefois, lorsque nous considérons le Web non comme une base de données statique, mais comme un réseau d'utilisateurs actifs, d'autres solutions au problème de la recherche et de la classification de ces informations se présentent.

En effet, les meilleures références à des sites ou à des documents utiles viennent souvent d'amis ou d'autres personnes partageant les mêmes intérêts. Le Web, et l'Internet en général, ne sont pas seulement des bases de données distribuées, mais nous offrent également la possibilité de construire et de gérer des réseaux sociaux à une échelle mondiale. Nous croyons que ces réseaux joueront un rôle très important dans l'exploitation des ressources sur l'Internet.

Nous ne sommes pas les seuls à émettre cette opinion. En particulier, des chercheurs à la Kyoto University, à NTT et au MIT Media Lab la partagent et ils sont déjà impliqués depuis plusieurs années dans la conception et la réalisation de systèmes exploitant des réseaux sociaux. Dans cette section, nous présentons brièvement une sélection de leurs systèmes, couramment connus sous le nom de ***Community Ware***. Nous nous concentrons sur des systèmes proches de FRIENDS Online, et en particulier sur Yenta, développé au MIT Media Lab. Nous présenterons FRIENDS Online lui-même dans la section suivante.

Hormis de la Kyoto University, de NTT et du MIT Media Lab, un grand nombre d'autres laboratoires sont impliqués dans la conception et dans l'implémentation du *Community Ware*. Par exemple, au sein du CNET, le Centre de Recherche et Développement de France Télécom, Cécile Bothorel travaille sur un système similaire à FRIENDS Online [Bothorel 98]. Toutefois, dans l'espace limité de cette thèse, nous ne pouvons présenter que les systèmes les plus connus et les plus pertinents.

Notons que le *Community Ware* est lié au *Group Ware*, mais qu'il existe des différences importantes. Dans le *Group Ware*, les groupes sont typiquement établis avant que le système ne soit activé et ils sont caractérisés par des buts communs. Par contre, dans le *Community Ware*, les groupes sont établis par le système même, et les membres de la population ne partagent pas nécessairement les mêmes objectifs.

Les travaux à la Kyoto University et à NTT

Toru Ishida, de la Kyoto University au Japon, est un des chercheurs les plus connus dans le domaine du *Community Ware* [Ishida 97]. Son livre *Community Computing, Collaboration over Global Information Networks*, catalogué également comme *Community Ware: Concepts and Practice*, offre un aperçu des travaux effectués par les chercheurs japonais, très productifs dans le domaine [Ishida 98a]. Notons qu'une partie importante de ces recherches sont réalisées au sein de laboratoires privés, notamment ceux de NTT et Sony.

Ishida identifie cinq fonctions différentes de *Community Ware* : il aide les utilisateurs à mieux se connaître, à partager des connaissances, à arriver à des consensus, à gérer la vie quotidienne et à organiser des événements sociaux. Dans le contexte de cette thèse, la deuxième fonction nous intéresse plus particulièrement, bien que les autres y sont très liées.

Le projet phare « ICMAS'96 Mobile Assistant Project », dirigé par des chercheurs de la Kyoto University, de NTT, du Nara Institute of Science and Technology et de la Kobe University, illustre les cinq fonctions identifiées par Ishida [Ishida 98b]. Il s'agit d'une expérience effectuée pendant la deuxième *International Conference on Multi-Agent Systems* (ICMAS'96), organisée à Kyoto en 1996 [Tokoro 96]. Au cours de cette expérience, une centaine d'Assistants Personnels Numériques (*Personal Digital Assistants* ou *PDA*) étaient prêtés aux participants, offrant le courrier électronique et l'accès à l'Internet, des informations touristiques, personnelles et liées à la conférence, et des services de forum et de prise de rendez-vous. D'un point de vue technologique, il s'agissait d'un projet ambitieux, expérimentant le Sony MagicLink, Telescript et MagicCap de General Magic, et la communication sans fil. D'un point de vue conceptuel, le projet était également révolutionnaire, incorporant un grand nombre de services différents.

Un de ces services est particulièrement intéressant dans le cadre de cette thèse : le *Community Viewer*. Le but de ce service est la visualisation et le renforcement de communautés d'utilisateurs ayant des intérêts communs. Le système tient compte de profils statiques et d'activités dynamiques. Dans l'expérience, les profils étaient des vecteurs de mots-clés rédigés par les utilisateurs. Dans les versions plus avancées du système, des vecteurs pondérés sont calculés automatiquement à partir d'informations déjà publiées par les participants sur le Web [Yoshida 98]. En analysant les relations entre les profils individuels, le système construit un plan en deux dimensions dans lequel des avatars représentant les utilisateurs sont positionnés. L'inspection de ce plan, personnalisé pour chaque participant, permet aux utilisateurs de retrouver des gens partageant les mêmes intérêts.

Une question importante concernant le *Community Viewer*, et tout système similaire est celle du respect de la vie privé (*privacy)*. En effet, ces systèmes ne peuvent être utiles que lorsque les utilisateurs permettent la publication, directe ou indirecte, d'informations personnelles. Nous présenterons notre point de vue de cette problématique dans notre discussion de FRIENDS Online.

L'expérience effectuée à ICMAS'96 illustre clairement les potentiels du *Community Ware*. Toutefois, elle a également illustré un certain nombre de problèmes. Une grande partie de ces problèmes étaient de nature technique, et trouvaient leur origine dans la nature *à la fois* ambitieuse et expérimentale de l'expérience. Par contre, la principale critique du système, comme elle était exprimée par les utilisateurs, était le fait que les niveaux de participation et d'activité n'étaient pas suffisants pour rendre le système intéressant. Autrement dit,

les gens ne s'engageaient pas parce que *les gens ne s'engageaient pas*. Nous reprendrons cette problématique importante et légèrement paradoxale dans notre discussion de FRIENDS Online.

Les travaux au MIT Media Lab

Un des groupes les plus actifs dans le développement du *Community Ware* est le Software Agents Group du MIT Media Lab, dirigé par Pattie Maes [software agents]. Le groupe approche la problématique des agents en soulignant le point de vue de l'utilisateur, ce qui est illustré par son utilisation des termes *personal assistant* et *interface agent* [Maes 94].

On peut soutenir que Maes et son équipe sont les inventeurs du principe de *collaborative filtering* [Lashkari 94]. Ce principe a été inspiré par deux problèmes liés à l'apprentissage par des agents assistants solitaires. Un tel agent apprend en observant et en analysant les actions de son utilisateur. A cause de cette approche, l'agent doit partir de zéro et ne peut pas apprendre des choses non montrées par l'utilisateur. Ces deux problèmes ont motivé Maes et al. à faire communiquer, et collaborer, plusieurs agents assistants travaillant pour des utilisateurs différents. Une des premières applications illustrant le principe de *collaborative filtering* était Firefly, un service pour la recommandation de disques. A l'heure actuelle, Firefly a été commercialisé [Firefly] et son principe peut être retrouvé dans un nombre important d'autres services telle que ceux offerts par Amazon.com [Amazon].

Les premières activités du groupe ont menées à toute une suite de projets, explorant tous - de manières différentes - le concept de *Community Ware*. Les tableaux 1a et 1b donnent un aperçu des projets actuels. Le symbole « ▲ » dans la colonne « C » indique que l'application est du type « Community Building and Referrals », la liant à FRIENDS Online. Un « ▲ » dans la colonne « MA » indique que l'application est basée sur le concept des Systèmes Multi-Agents. Pour plus de détails, se référer à [software agents].

Nom	**Fonction — Description — Références**	**C**	**MA**
Amalthea	la découverte et le filtrage d'informations par des agents coopératifs et concurrents [Moukas 96]		▲
Butterfly	la recommandation de groupes conversationnels [Van Dyke 99]	▲	
Collaborative Ontology Development	le développement d'ontologies utilisées par des groupes d'utilisateurs de grande taille	▲	
Expert Finder	la localisation d'experts dans la communauté	▲	▲
Footprints	la recherche d'informations en suivant les traces des autres utilisateurs	▲	
Friend of a Friend Finder	l'exploitation de réseaux sociaux afin de trouver des informations	▲	▲
Kasbah	l'automatisation de transactions commerciales à l'aide d'agents [Chavez 96]		▲
Let's Browse	la navigation collaborative du Web	▲	

Tableau 1a : Projets du Software Agents Group au MIT Media Lab (partie 1).

Nom	Fonction — Description	C	MA
Letizia	la navigation sur le Web assistée par un agent [Lieberman 95]		
Market Maker	la personnalisation de marchés virtuels — extension de Kasbah		▲
Mobile Agents for Routing Discovery	la construction de cartes de la topologie de réseaux à l'aide d'agents mobiles		▲
PDA@Shop	les achats économiques à l'aide d'agents mobiles		▲
Remembrance Agents	l'aide à la mémoire associative [Rhodes 96]		
Reputation Mechanisms	le contrôle de fiabilité des marchands dans un marché électronique (comme Kasbah)		▲
Straum	la représentation d'utilisateurs sur l'Internet par une écologie d'agents [Minar 98]	▲	▲
Tête-à-tête	la facilitation des transactions commerciales en couplant les consommateurs aux marchands [Guttman 98]		▲
Trafficopter	la collection et la communication d'informations routières		▲
Yenta	la construction de groupes d'utilisateurs partageant des intérêts [Foner 97b]	▲	▲

Tableau 1b : Projets du Software Agents Group au MIT Media Lab (partie 2).

Quatre de ces applications sont du type « Community Building and Referrals » tout en appliquant le concept des Systèmes Multi-Agents. Il s'agit de : Expert Finder, Friend of a Friend Finder, Straum et Yenta. De ces quatre, Yenta est la seule groupant explicitement les utilisateurs, comme le fait FRIENDS. Etudions de plus près son fonctionnement.

Yenta

Yenta est un système conçu par Leonard Foner, permettant une population d'agents de se grouper et de partager des informations comme une *computational ecology* [Foner 95] [Foner 96a] [Foner 97b]. Dans une *computational ecology*, les agents n'ont que des connaissances locales, et s'auto-organisent en entités d'un niveau supérieur [Huberman 88].

Bien que les idées à la base de Yenta sont d'une nature générale, l'application même est conçue pour marier des utilisateurs ayant des intérêts similaires. Le système est de nature distribuée pour éviter les problèmes liés à une approche centralisée. Foner en identifie deux. D'abord, un système centralisé souffre nécessairement du problème de *scaling*. De plus, il est plus vulnérable au niveau du fonctionnement technique et au niveau de la confidentialité des données. En effet, lorsqu'un système centralisé s'effondre, plus rien ne fonctionne. D'une manière similaire, lorsque quelqu'un d'extérieur accède à un système centralisé, toutes les données sont compromises. Notons que les aspects de sécurité jouent un rôle important dans ces travaux [Foner 96b].

Foner propose un mécanisme particulier pour implémenter la fonctionnalité de *matchmaking* d'une manière distribuée. Il rejette l'approche centralisée, les méthodes trop probabilistes, et la construction et l'utilisation d'arborescences hiérarchiques, argumentant que « …hierarchical, single-inheritance trees are often used for such coordination, but there is no obvious a priori hierarchy in this application by which to organize the agents (why would any one person's interests be at the top of any hierarchy? how would we know whom to pick, anyway?) » [Foner 97b]. Nous répondrons à cette question dans la section suivante. Dans un premier temps, continuons notre étude de Yenta. Dans son approche, Foner s'inspire des idées de *computational ecology*. En particulier, il propose :

— de comparer les informations des agents d'une manière décentralisée : d'agent à agent,

— d'utiliser des références d'un agent à un autre et des algorithmes du type *hill-climbing* pour trouver des agents plus similaires, dans le but de

— construire des groupes d'agents similaires, et

— d'utiliser ces groupes pour présenter des agents entre eux et pour faciliter la communication entre des utilisateurs ayant des intérêts similaires,

— d'utiliser des agents persistants, tournant pendant de longues laps de temps : le processus de regroupement nécessite du temps

En ce qui concerne l'implémentation concrète, quatre questions importantes se posent. Comment peut-on définir et établir les intérêts des utilisateurs ? Comment peut-on déterminer la similarité entre intérêts ? Comment les agents doivent-ils explorer leur environnement social ? Comment peut-on former des groupes d'agents similaires ?

Dans Yenta, les intérêts des utilisateurs sont déduits de leurs documents électroniques. Spécifiquement, les articles *newsgroup* et les messages *email* sont considérés. Chaque document est traité comme une unité d'information appelée un **grain**. Un utilisateur est considéré avoir un intérêt particulier pour quelque chose lorsque, entre ses documents, se trouvent plusieurs grains similaires. Une collection de tels grains est appelée un **granule**. Un profil d'un utilisateur contient typiquement plusieurs granules différents. Deux utilisateurs différents forment un **groupe** si leurs profils contiennent des granules similaires. La figure 1 illustre les relations entre agents, grains, granules et groupes.

La similarité entre grains, limités - dans la version actuelle de Yenta - à des documents de texte, est calculée en utilisant la mesure *term-frequency x inverse-document-frequency*, présentée dans la section précédente. Comme mesure de similarité entre deux documents, Yenta utilise le produit scalaire des deux vecteurs correspondants.

Au sein des agents individuels, des grains différents sont combinés en granules par un processus appelé *preclustering*. L'algorithme utilisé est de complexité $O(n^2)$, mais ne doit être exécuté qu'une fois pour chaque agent. Des grains additionnels peuvent être ajoutés d'une manière graduelle. Une fois les granules formés, les agents essaient de trouver des agents similaires. Trouver le premier agent ressemblant constitue un problème particulier et nécessite un processus de *bootstrapping*. Dans Yenta, ce processus est basé sur un certain nombre d'heuristiques.

Une fois qu'un agent en a trouvé un autre, les granules des deux peuvent être comparés et des groupes peuvent être formés. Un groupe est crée lorsque la similarité entre deux granules excède un seuil donné. Chaque agent intègre la description des groupes formés dans son ***cluster cache***, une structure de données locale, décrivant les groupes dont l'agent fait partie. Puis, chaque agent inspecte le *rumor cache* de l'autre. Le ***rumor cache***, une deuxième structure locale, contient des informations sur d'autres agents, pas pertinentes pour le propriétaire du *cache*, mais potentiellement pertinentes pour des agents que le propriétaire pourrait rencontrer. Les informations pertinentes dans le *rumor cache* d'un agent sont ajoutées au *cluster cache* de l'autre. Le reste est ajouté au *rumor cache*. Lorsque la taille limite des *caches* est atteinte, les informations les plus anciennes sont éliminées.

Une fois les caches mises à jour, les agents contacteront les agents non encore visités dans leur cluster cache, et le cycle recommence. La liste des agents non encore contactés s'appelle le ***pending-contact list***. En général, les références aux autres agents dans les caches sont appelées ***referrals***, et leur exploitation est à la base du fonctionnement de Yenta.

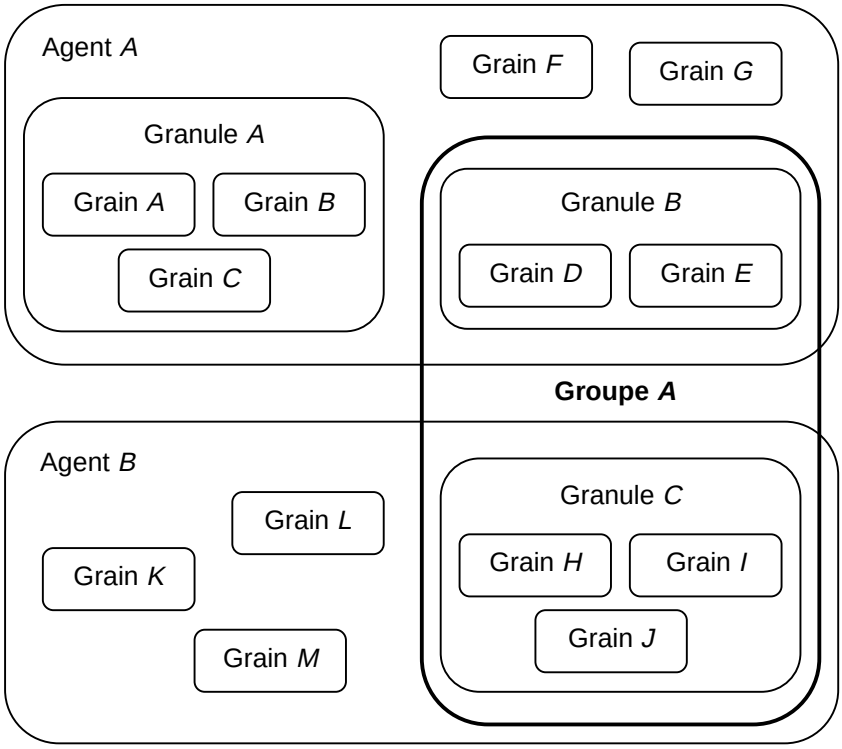


Figure 1 : Structures dans Yenta.

Foner n'a pas encore testé Yenta à une échelle « grandeur nature », i.e. en utilisant une grande population d'utilisateurs. Toutefois, en simulation, les résultats sont encourageants. Il s'agit d'un projet important sur lequel un nombre respectable d'étudiants travaillent au MIT. Son but est lié de près à celui de FRIENDS Online, bien que son approche et son fonctionnement soient sensiblement différents. Nous comparerons les deux systèmes plus loin.

Un Web auto-organisant

Des systèmes comme Yenta sont des systèmes auto-organisants : ils ne contiennent pas d'entité centrale organisant les constituants. On peut se poser la question de savoir si, au lieu de construire une couche au-dessus du Web, on ne peut pas rendre le Web même auto-organisant.

A la périphérie du domaine du *Community Ware*, un certain nombre de chercheurs proposent des mécanismes associatifs permettant l'auto-organisation du Web en un réseau de concepts. Dans un contexte cybernétique, l'introduction de tels mécanismes peut mener à l'émergence d'un « cerveau global » [Heylighen 96].

Cette auto-organisation du Web, couplant topologie et informations, peut faciliter énormément la recherche d'informations sur le Web. Par contre, la plupart des travaux effectués sont surtout théoriques et exigent une modification globale du Web. Une telle modification est difficile à réaliser à cause de la nature presque autonome de l'Internet.

Dans cette thèse, nous nous concentrons sur des systèmes plus réalistes, et nous ne détaillerons pas les recherches sur l'auto-organisation du Web. Notons toutefois que des mécanismes concrets ont été développés et testés à une échelle locale, entre autres par Johan Bollen et Francis Heylighen [Bollen 96] [Bollen 97] et l'équipe de Paul Francis à NTT [Ingrid].

IV.2.3 FRIENDS Online

Principe

FRIENDS Online est une version de FRIENDS conçue pour l'utilisation sur l'Internet. En gérant l'espace virtuel de ses utilisateurs, l'application peut faciliter des activités tels que la recherche et la classification d'informations sur l'Internet. Détaillons son fonctionnement du point de vue de l'utilisateur.

L'idée de base de FRIENDS Online est la suivante : se faire représenter sur l'Internet par un agent mobile migrant de site à site à la recherche d'agents représentant des utilisateurs ayant des intérêts similaires. FRIENDS Online se distingue des applications similaires, tel que Yenta, par le fait que la population d'agents constitue un SMAM augmenté. En conséquence, tous les agents sont groupés d'une manière récursive et binaire.

Lorsqu'un utilisateur souscrit au service, le système lui demande un alias, un mot de passe et une liste de mots-clés d'une taille minimale, par exemple 10. Eventuellement, le système peut suggérer des synonymes ou des manières d'écriture alternatives pour minimiser les malentendus. En effet, une fois les mots-clés entrés dans le système, tout calcul de similarité et tout regroupement d'agents est basé sur ces chaînes de caractères. Dans aucun cas, le système tente d'interpréter les mots-clés. Par contre, il peut assister un maximum l'utilisateur dans les processus de rédaction et de modification éventuelle de sa liste.

Le choix des mots-clés est essentiel, car ils constituent le profil de l'utilisateur. Yenta, et beaucoup d'autres systèmes, les déduisent automatiquement. Toutefois, nous croyons que leur choix est trop important pour le laisser à des algorithmes. Comment décrire ses intérêts (ou soi-même) avec uniquement dix mots-clés n'est pas une tâche aisée. En effet, nous croyons qu'elle est assez difficile, et donc intéressante, et peut motiver les gens à s'y intéresser.

La version actuelle de FRIENDS Online utilise des vecteurs binaires de mots-clés. On peut imaginer des versions plus sophistiquées utilisant des vecteurs pondérés, ou des éléments d'une nature très différente, tels que des sons ou des couleurs. En général, nous croyons que l'aspect multimédia des applications informatiques ne cessera pas de gagner en importance.

Une fois qu'un utilisateur a rédigé sa liste de mots-clés, le système crée pour lui un agent atomique ayant cette liste, son alias, et son mot de passe encrypté comme attributs. L'agent est créé sur le site auquel l'utilisateur est connecté. FRIENDS Online est conçu pour s'accommoder de millions d'utilisateurs et toute implémentation « grandeur nature » est nécessairement d'une nature distribuée. On peut imaginer que l'utilisateur typique se connecte au site topologiquement le plus proche de lui.

La philosophie de base de FRIENDS Online est d'attribuer un seul profil, et donc un seul agent, à chaque utilisateur. On peut imaginer des systèmes dans lesquels chaque utilisateur se fait représenter par plusieurs agents, par exemple un agent « professionnel » et un agent « loisirs ». Toutefois, nous croyons que c'est exactement l'exploitation de corrélations implicites qui rendent le concept intéressant. Dans un système comme Yahoo, l'utilisation d'un grand nombre de *cross-links* dilue la classification. D'une manière similaire, dans FRIENDS Online, l'introduction de plusieurs profils différents par utilisateur diminue la puissance de classification du système.

Une fois l'agent atomique créé, il est intégré dans le SMAM local au site. Plus spécifiquement, il est couplé avec l'agent le plus similaire qui est disponible. Plusieurs tactiques différentes peuvent être suivies pour intégrer le nouvel agent dans le SMAM local. En tout cas, il fera partie d'un couple représenté par un agent composé faisant partie - lui aussi - d'un groupe plus important (sauf s'il s'agit de l'agent représentant la population complète).

Dès que l'agent est intégré, il commence la recherche d'un meilleur partenaire, sauf s'il se trouve couplé à un agent identique. Il s'agit d'un processus continu exécuté par tous les agents, atomiques ou composés. La méthode suivie par un agent peut être très simple, comme celle d'un agent réactif. Toutefois, elle peut aussi être très sophistiquée, impliquant les processus de planification et de communication d'un agent cognitif. Dans FRIENDS Online, les stratégies peuvent différer d'un agent à l'autre. En termes de cycles de calcul, un agent cognitif est typiquement plus coûteux qu'un agent réactif. On peut imaginer des versions de FRIENDS Online dans lesquels les utilisateurs sont facturés selon les cycles consommés par leurs agents. Un tel principe stimulera sans doute le développement d'agents efficaces.

Du point de vue de l'utilisateur, une première fonction de base de FRIENDS Online consiste en le suivi de son agent dans le SMAM. Dans le temps, son agent se trouvera typiquement entouré par des agents de plus en plus similaires. Selon les services secondaires offerts par FRIENDS Online, l'utilisateur peut directement contacter les personnes représentées par ces agents, ou - par exemple - inspecter leur Web *bookmarks*. Afin d'assurer une exploitation optimale de cette fonctionnalité, la visualisation du SMAM autour l'agent de l'utilisateur doit être - du point de vue ergonomique - d'une grande qualité.

Au niveau technique, l'utilisateur peut se connecter à n'importe quel site pour trouver son agent. Si l'agent ne réside pas au site contacté, le site propagera la recherche aux sites voisins. Alternativement, l'agent d'un utilisateur peut le prévenir chaque fois qu'il change de site.

La deuxième fonction de base de FRIENDS Online consiste en l'inspection de la classification automatique effectuée par le système. En suivant les mots-clés associés à des agents composés, et donc à des groupes, des groupes et des utilisateurs spécialisés peuvent être trouvés. Au-dessus du système de base, une fonctionnalité de forum peut être implémentée permettant d'adresser des questions, d'une manière publique (par *broadcast*) à tous les membres d'un groupe donné.

L'inspection de la classification peut être utile non seulement pour les participants à FRIENDS Online, mais également pour des utilisateurs externes au système. Toutefois, il y a des questions de confidentialité qui se posent. En publiant leur profil, les participants à FRIENDS Online sacrifient une petite partie de leur *privacy* afin de profiter des services offerts. Il ne serait pas correct d'offrir ces services à des gens non engagés dans le système. Par conséquent, la meilleure stratégie nous semble être de laisser le système inaccessible aux gens qui n'y participent pas (en offrant un profil). Notons que la nature fermée du système peut *à la fois* intriguer et frustrer les gens et les motiver à participer.

La classification effectuée par FRIENDS Online peut relever des corrélations implicites. Prenons l'exemple de six utilisateurs ayant - respectivement - comme mots-clés : {animaux, ordinateurs, PC}, {Macintosh, opéra, ordinateurs, voyages}, {ordinateurs, UNIX, Web}, {cryptographie, ordinateurs, UNIX}, {Macintosh, ordinateurs, vins}, {ordinateurs, PC, photographie}. Une fois le système stabilisé, il aura créé les groupes « Macintosh », « PC » et « UNIX » comme sous-groupes de « ordinateurs ». Notons que ce type de classification ne nécessite aucune forme d'interprétation de la part du système.

Lorsqu'un utilisateur suit son agent ou inspecte la classification, il peut se rendre compte que - dans le contexte de la population donnée - son choix de mots-clés ne reflète pas correctement ses intérêts. Très concrètement, ceci implique que son agent personnel ne se trouve pas dans les groupes avec lesquels l'utilisateur peut ou veut s'associer. Une terminologie mal adaptée à la population peut être à la base de ce problème. Par exemple, si la plupart des participants utilisent le mot-clé « Web » et un seul utilisateur utilise le terme « WWW », il est dans son intérêt de s'adapter au vocabulaire de la population et de changer son mot-clé (sauf s'il ne veut pas faire des concessions de cette nature).

Il est donc très important qu'un utilisateur puisse facilement modifier ses mots-clés. Cette fonctionnalité lui permet de faire naviguer son agent dans l'espace social de SMAM. Sans cette boucle rétroactive, FRIENDS Online ne peut pas atteindre son potentiel.

FRIENDS Online et Yenta

FRIENDS Online ressemble beaucoup à Yenta, mais il existe des différences importantes. Tout d'abord, FRIENDS Online exige plus d'efforts de la part des utilisateurs : ils doivent participer d'une manière plus active dans l'auto-organisation du système. Toutefois, ceci ne nous semble pas gênant. D'une part, nous croyons que la nature interactive du système peut être vécue d'une manière positive. D'autre part, nous sommes sceptiques à propos de la construction automatique de profils.

Par contre, la différence principale entre les deux systèmes est le fait que les groupes sont structurés dans FRIENDS Online et plats dans Yenta. Dans la section précédente, nous avons présenté les objections de Foner : il ne voit pas de critères rationnels pour imposer une structure hiérarchique. Il nous semble que cette analyse est liée au fait que, dans la classification classique, un mot-clé est attribué à chaque nœud de l'arbre de

classification. Toutefois, comme nous l'avons déjà souligné, un SMAM n'est pas un arbre binaire. Les « nœuds », « feuilles » exclues, représentent des *groupes* et si les sous-groupes n'ont pas de mots-clés en commun, aucun mot-clé n'y est associé. En ce qui concerne la priorité des mots-clés, il nous semble assez naturel que les mots-clés partagés par un grand nombre d'utilisateurs prennent une place haute dans la hiérarchie et que les mots-clés distinguant des utilisateurs similaires se trouvent plus bas.

Lorsqu'on se limite - comme nous le conseillons - à un seul profil par participant, chaque utilisateur se trouve à un endroit précis dans l'espace social, même s'il a des intérêts non représentés dans son environnement local, mais toutefois représentés ailleurs dans le SMAM. En général, chaque organisation est un compromis et la fréquence des mots-clés détermine leur importance dans la classification. Cette nature stricte du système ne nous gène pas. Au contraire, elle nous semble très naturelle. Par exemple, l'espace physique dans lequel nous vivons a exactement la même propriété : nous ne pouvons pas être à deux places en même temps. Il est vrai que nous voyageons beaucoup dans l'espace physique et il nous semble logique que nous voulons également voyager dans FRIENDS Online. Pour l'instant, ceci est possible en modifiant directement les mots-clés. Dans de futurs versions plus sophistiquées, nous pouvons envisager des mécanismes plus conviviaux.

Dans le chapitre précédent, nous avons indiqué la dépendance de la qualité de l'organisation des agents de la nature de la population. Par exemple, si les membres d'une population n'ont aucun mot-clé en commun, leur structuration - en ce qui concerne l'harmonie - est arbitraire, car l'harmonie est toujours égale à 0. Evidemment, même dans ce cas, l'entropie du SMAM augmenté est optimisée. Toutefois, nous croyons que - dans un contexte concret - des situations de ce type sont exceptionnelles, surtout dans un système opérationnel pendant un certain période de temps. En effet, nous posons comme hypothèse que le système global, utilisateurs inclus, évolue d'une telle manière que les utilisateurs s'approchent les uns des autres en adoptant les mêmes mots-clés, tout en se distinguant par l'introduction de nuances de ce qu'ils ont en commun. Cette hypothèse ne fait pas partie de cette thèse dont le sujet est l'informatique plutôt que la sociologie ou la psychologie sociale. Toutefois, elle est directement liée à la philosophie étant à la base du concept des SMAM.

En effet, le thème central de l'*Art of Insanity*, introduisant cette philosophie, est l'évolution des cultures existantes vers la Culture Universelle [Van Aeken 94]. Cette évolution est vue comme positive : plus on se ressemble, plus on peut développer des nuances. Non par coïncidence, elle correspond à l'évolution naturelle d'un SMAM représentant la population. Illustrons le principe à l'aide du SMAM de la figure 2 : le SMAM (population) *A* est composé des deux agents (membres) *B* et *C*. Les deux agents se ressemblent, mais ils sont différents. A cause de leur composition, ils ne peuvent pas être parfaitement équilibrés. Mais, en interagissant, ils peuvent s'équilibrer. Par exemple, la migration de l'agent *D* vers l'agent *E* équilibrera *B*, *C* et *A*. La théorie des SMAM garantit une telle évolution.

Selon un mécanisme similaire, nous croyons, sans le démonter dans le cadre de cette thèse, que des populations réelles - dans un contexte fermé - essaient de trouver l'équilibre maximal. Dans le cadre de FRIENDS Online, ceci implique que les utilisateurs essayeront, jusqu'à un certain degré, d'établir les mêmes mots-clés.

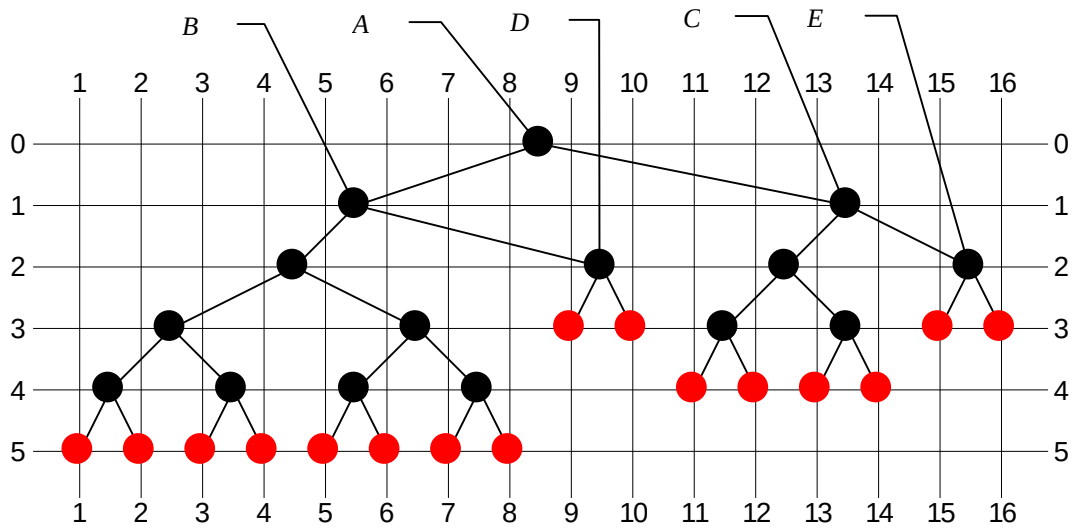


Figure 2 : La migration de *D* vers *E* équilibrera le système.

L'espace social généré par Yenta est moins structurée que celui des SMAM augmentés. Sa nature plus libre implique certains avantages, mais également un certain nombre d'inconvénients. Par exemple, au niveau conceptuel, dans une population composée de deux utilisateurs partageant 10 mots-clés (granules), Yenta crée 10 groupes, plutôt qu'un seul groupe comme le fait FRIENDS Online.

En ce qui concerne l'analyse scientifique des deux systèmes, FRIENDS Online est, dans certains aspects, plus simple que Yenta et plus facile à modéliser. En effet, au cœur de l'application se trouve le modèle des SMAM augmentés, lié de près au modèle des SMAM, extrêmement simple. Par conséquent, nous pouvons appliquer les outils théoriques que nous avons développés, tel que la mesure d'entropie. Par contre, à la base de Yenta ne se trouve pas un modèle étudié hors du contexte de l'application. Ses mécanismes sont élégants et logiques, mais *ad hoc*, et ils ne facilitent pas nécessairement son analyse.

Dans FRIENDS Online, nous avons fait abstraction des algorithmes utilisés par les agents. Nous n'avons spécifié le comportement des agents qu'au niveau conceptuel. Dans ce sens, notre système est moins bien spécifié que Yenta. Evidemment, pour les implémentations concrètes de l'application, nous avons développé des algorithmes concrets, permettant une analyse plus spécifique. Comme nous l'avons souligné dans les chapitres II.3 et IV.1, il ne s'agit pas d'un problème trivial, mais d'un problème très « IA » que nous avons proposé, dans le contexte des SMAM purs, comme un *benchmark*.

Nous croyons que FRIENDS Online et Yenta sont tous les deux caractérisés par certains avantages et inconvénients. Les deux systèmes suivent une philosophie différente, et une comparaison absolue n'est pas possible. Selon les circonstances, l'utilité relative des deux peut varier.

Implémentation

Nous avons implémenté deux versions de FRIENDS Online : une version centralisée de qualité « industrielle » et une version distribuée de nature expérimentale. Daniel Genovese, travaillant dans notre équipe comme stagiaire CNAM (Conservatoire National des Arts et Métiers), est le réalisateur principal de ces deux applications. Son mémoire CNAM présente leur description détaillée [Genovese 98]. Dans le reste de cette section, nous présenterons brièvement FRIENDS Online 3.1, la version centralisée la plus récente. Le manuel

d'utilisateur de cette application se trouve en annexe. Dans la dernière section de ce chapitre, nous présenterons également - sans aller dans les détails - FRIENDS Online MAI, la version distribuée du système.

FRIENDS Online 3.1 est une application client - serveur écrite en Java [Java]. Le serveur est une application indépendante, plutôt qu'une applet. Les clients sont des applets qui peuvent également être exécutées comme des applications. Le système utilise le JDK 1.1 (*Java Developers Kit* 1.1), nécessitant des versions récentes des navigateurs Web pour exécuter les applets.

Sachant la nature de FRIENDS Online, le choix de Java comme langage de programmation était logique. Simple, moderne, complet et adapté à une utilisation sur l'Internet, il nous a facilité sensiblement le développement. L'utilisation du compilateur JIT (*Just In Time*) de Symantec nous a permis d'obtenir une performance acceptable. Comme environnement de développement, nous avons utilisé Symantec Café, le prédécesseur de Visual Café [Café].

Nous avons développé deux types différents de clients : un type « utilisateur » et un type « administrateur ». Les figures 3 et 4 montrent des copies d'écran des fenêtres principales de ces deux aspects. Les intérêts visualisés sont d'une nature fictive, bien que les noms des utilisateurs coïncident avec ceux des membres de notre équipe.

Un utilisateur se connecte au serveur, *soit* en chargeant du site serveur la page contenant l'applet « utilisateur », *soit* en exécutant l'applet - déjà chargée - comme application (ce qui nécessite la spécification du site serveur). Une fois connecté, le serveur lui demande son alias et son mot de passe. Dans la version 3.1, les mots de passe ne sont pas encryptés. Si le serveur ne connaît pas l'utilisateur, il lui demande de spécifier une liste de mots-clés. Une fois la liste rédigée, un agent atomique représentant l'utilisateur est créé. Dès que l'utilisateur est identifié, ou enregistré, la fenêtre principale est affichée.

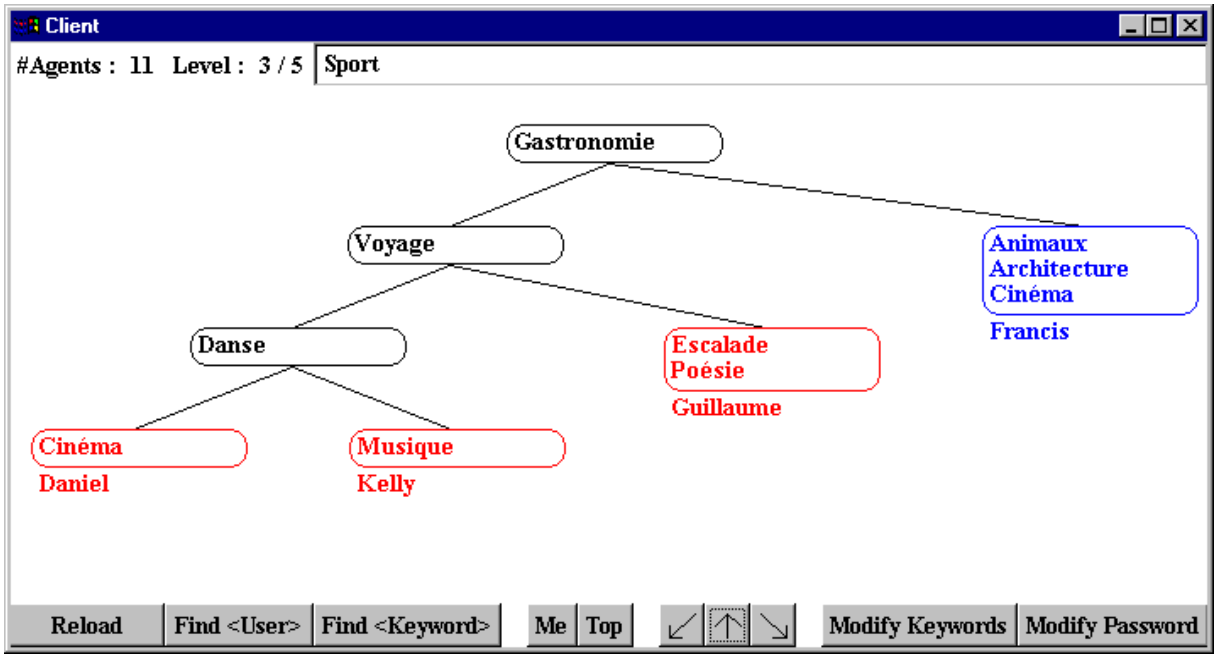


Figure 3 : Fenêtre principale de FRIENDS Online « Client ».

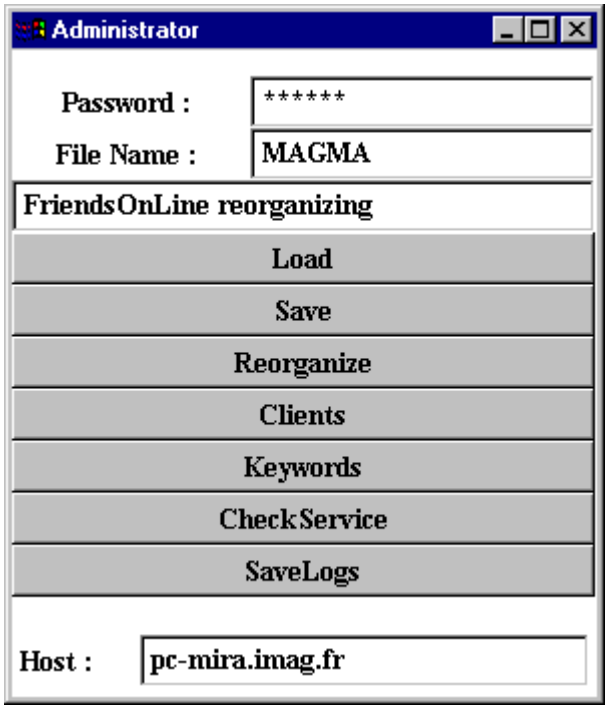


Figure 4 : Fenêtre principale de FRIENDS Online « Administrator ».

Les fonctionnalités principales du client « utilisateur » sont la **navigation**, la **recherche** d'utilisateurs ou de mots-clés et la **modification** des données personnelles. A chaque moment, le client contient une copie du SMAM global géré par le serveur. La copie est mise à jour lorsque l'utilisateur effectue certaines opérations. S'il le souhaite, l'utilisateur peut également forcer la mise à jour.

L'utilisation du client « administrateur » nécessite un mot de passe particulier. Une fois l'authentification effectuée, l'administrateur peut **sauvegarder** et **charger** des SMAM, **forcer** la **réorganisation globale**, **travailler** sur les listes de **mots-clés** et d'**utilisateurs** et **enregistrer** les ***logs*** des activités du système.

Notons que le manuel de FRIENDS Online 3.1 présente une description complète de toutes les fonctionnalités des deux clients.

Au cœur de FRIENDS Online 3.1 se trouve le protocole de communication entre le serveur et les clients. Ce protocole est détaillé dans [Genovese 98]. Deux algorithmes différents sont utilisés pour réorganiser le SMAM. La réorganisation n'est pas continue : elle n'est effectuée que lorsqu'il y a des changements extérieurs ou lorsque l'administrateur la demande. C'est une des différences entre cette application et FRIENDS Online MAI. Dans la deuxième, la réorganisation est continue.

La réorganisation globale, initiée par l'administrateur, utilise le même algorithme que FRIENDS Numbercruncher, présenté dans le chapitre précédent. Un autre algorithme est utilisé pour intégrer, d'une manière aisée, de nouveaux utilisateurs ou des utilisateurs ayant changé leur liste de mots-clés. L'algorithme est le suivant :

1) Chercher, dans le SMAM, l'agent le plus ressemblant au nouvel agent. Seuls les agents ressemblant moins à leur partenaire qu'au nouvel agent sont considérés.

```
2) Coupler le nouvel agent à l'agent trouvé. Le nouveau couple
   remplace l'agent original.
```

Notons que la complexité de l'algorithme est, pour une seule intégration, de l'ordre $O(n)$.

IV.2.4 Evaluation

Nous avons soumis FRIENDS Online 3.1 à un grand nombre de tests et nous sommes confiants en sa fiabilité.

Nous avons également effectué une expérience à une échelle plus importante. Pendant la conférence ICMAS'98, organisée à Paris [Demazeau 98], nous avons offert le service de FRIENDS Online 3.1 aux participants. 50 personnes des 500 présents ont participé à l'expérience. Au niveau technique, le serveur s'est avéré 100% fiable : fonctionnant à Grenoble, il était actif - sans interruption - pendant toute la durée de la conférence (5 jours). Côté clients, nous n'avons rencontré aucun problème. Au niveau conceptuel, les utilisateurs étaient enthousiastes. Toutefois, le nombre de points d'accès était limité, ne permettant que peu d'interactions entre utilisateurs et système.

La courte durée de l'expérience, combiné avec l'interactivité limitée et le nombre modeste d'utilisateurs n'ont pas permis une évaluation complète au niveau conceptuel. De plus, une telle évaluation implique la participation de chercheurs en psychologie, en sociologie et en sciences de l'information : elle dépasse le contexte de cette thèse en informatique. Toutefois, cette expérience a montré la faisabilité du concept et la fiabilité de son implémentation.

L'expérience a également illustré quelques problèmes. Par exemple, un certain nombre d'utilisateurs avaient la tendance à spécifier - comme « mots-clés » - de longues chaînes de caractères comptant plusieurs mots, difficiles à visualiser dans nos applets. Un problème plus sérieux était la participation limitée. Selon la demande des organisateurs de la conférence, la promotion du système était modeste, et il est probable qu'un certain nombre des participants n'étaient pas avertis de son existence. De plus, en général, les systèmes tels que FRIENDS Online ne séduisent les gens que lorsqu'ils comptent déjà un grand nombre d'utilisateurs [Ackerman 96]. Dans la première phase de leur opération, ces systèmes sont peu populaires et nécessitent la promotion externe. Par contre, une fois la masse critique atteinte, ils attirent plus facilement de nouveaux utilisateurs.

Nous sommes prêts pour tester FRIENDS Online 3.1 à une échelle plus grande, dans l'espace et dans le temps. Une fois que les versions populaires des navigateurs supporteront à 100% le JDK 1.1, le service pourra être ouvert à une sélection arbitraire des millions d'utilisateurs sur l'Internet. Toutefois, FRIENDS Online 3.1 utilise un serveur centralisé, limitant son application à quelques milliers d'utilisateurs. Sa réécriture en C++ peut augmenter sa capacité d'un facteur fixe, mais ne résoudra pas le problème de *scaling*. Pour ces raisons, nous avons expérimenté une version distribuée de l'application.

Avant de présenter cette version, soulignons que FRIENDS Online 3.1 n'implémente que l'essentiel du concept de FRIENDS Online. Une version commerciale de l'application doit offrir une interface graphique plus ergonomique. En particulier, la composition de groupes ne doit pas être présentée d'une manière binaire, mais d'une manière plus adaptée au système visuel humain. De plus, afin de vraiment exploiter le potentiel du concept, des services secondaires doivent être offerts au-dessus du service de base. En particulier, les utilisateurs bénéficieront de mécanismes permettant de partager des informations (tel que des *bookmarks*) et de

communiquer (au sein d'un groupe donné). Selon le sujet des communications, le groupe dans lequel les messages sont distribués peut varier d'une seule personne à la population complète.

IV.2.5 FRIENDS Online MAI

Une version de FRIENDS Online supportant plus que quelques milliers d'utilisateurs doit nécessairement être distribuée. Deux approches différentes se présentent : l'approche classique basée sur l'utilisation d'un serveur distribué et l'approche plus récente utilisant la technologie des agents mobiles.

FRIENDS Online est un système basé sur la notion des agents, et il se prête naturellement à une implémentation distribuée à l'aide d'agents mobiles : il suffit d'implémenter chaque agent comme un agent mobile. De plus, cette technologie permet l'utilisation d'algorithmes différents au sein des agents différents. L'implémentation d'un tel SMAM hétérogène en utilisant la technologie classique (client - serveur distribué) est beaucoup plus délicate.

Comme plate-forme d'agents mobiles, nous avons retenu les Aglets d'IBM [Aglet]. A l'heure actuelle, cette plate-forme est une des plus connues. De plus, elle se fonde à 100% sur Java et est conçue explicitement pour l'implémentation d'agents. Ces raisons nous ont motivé pour l'adopter.

Nous n'avons implémenté qu'une partie des fonctionnalités de FRIENDS Online. De plus, la version actuelle n'a pas été soumise aux tests et aux contrôles intensifs utilisés pour vérifier FRIENDS Online 3.1. En effet, en construisant FRIENDS Online « Mobile Agent Implementation », nous avons testé la *faisabilité* d'une implémentation basée sur la technologie d'Aglets. Le prototype actuel est expérimental et il n'est pas conçu pour sortir du laboratoire.

Une description détaillée du prototype est présentée dans [Genovese 98]. Il permet la migration physique des SMAM, tout en supportant la réorganisation *continue* des agents. Le fait que la réorganisation est continue exige un protocole de migration très strict. Du point de vue théorique, ce protocole constitue l'élément le plus intéressant de FRIENDS Online MAI. Concluons ce chapitre avec sa présentation.

Le protocole de migration

Si le parallélisme des agents d'un SMAM est réel plutôt que simulé par un processus central, un protocole de migration est nécessaire non seulement pour des raisons conceptuelles, mais également pour des raisons techniques.

Etudions l'exemple de la figure 5 : *A* veut migrer vers *B*. En même temps, *C* veut migrer vers *D*. Evidemment, une fois *C* migré, *B* n'existe plus. Si l'agent *A* arrive, après que *C* a migré, il sera perdu et l'intégrité du système sera compromise.

Un autre problème se pose lorsque deux partenaires décident de migrer en même temps. Supposons que les agents *A* et *B* de la figure 6 décident de migrer en même temps (vers *D* et *E*, respectivement). Ceci impliquera deux fois la destruction de *C*, ce qui, évidemment, n'est pas possible.

Pour éviter ce genre de problèmes, nous sommes obligés d'utiliser un protocole de migration tel que le protocole général présenté dans le chapitre III.3. Là où le protocole général est purement conceptuel, le protocole que nous présentons ici est de nature technique et *doit* être implémenté si la concurrence des agents est réelle.

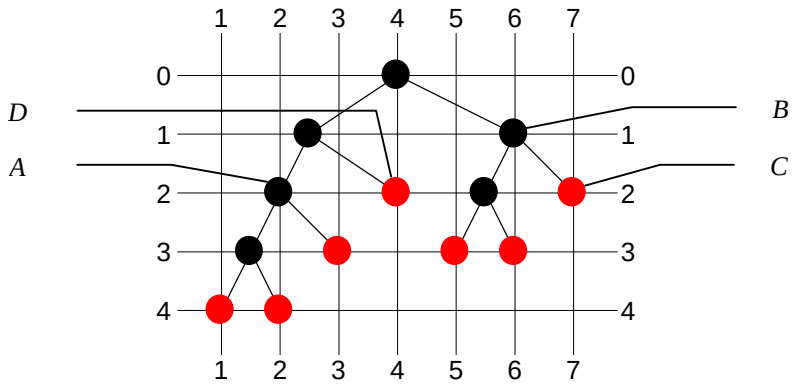


Figure 5 : Migration simultanée d'*A* vers *B* et de *C* vers *D*.

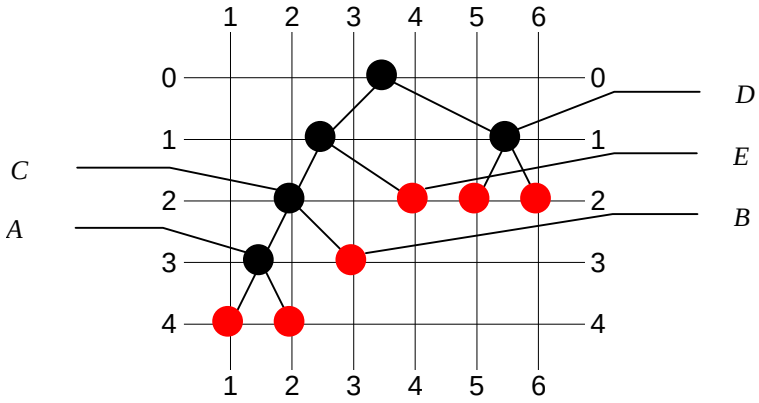


Figure 6 : Migration simultanée d'*A* vers *D* et de *B* vers *E*.

Supposons que chaque agent est implémenté à l'aide de deux *threads*. (Nous utilisons le terme « thread », plutôt que le terme « processus léger » pour souligner que, par (notre) définition, à chaque agent correspond, au moins du point de vue conceptuel, exactement un processus.) Le premier *thread* implémente le comportement « qui se ressemble, s'assemble ». Le deuxième *thread* s'occupe des messages reçus par l'agent. (Cet arrangement correspond essentiellement à l'architecture que nous avons utilisée dans FRIENDS Online MAI, où chaque Aglet compte deux *threads*.)

La clé du protocole consiste en la demande de permission de migration de la part de l'agent migrateur *à la fois* au super-agent et au **destinataire** (l'agent avec lequel l'agent migrateur veut se coupler). La variable booléenne `libre` indique si l'agent migrateur est engagé (comme destinataire ou comme migrateur). S'il est engagé, `libre` est à faux. La variable booléenne `filsMigrant` indique si un des fils de l'agent est un train de migrer. Si c'est le cas, `filsMigrant` est à vrai. L'algorithme est le suivant :

```
THREAD « qui se ressemble, s'assemble »

{
    libre := vrai
    Répéter indéfiniment {
```

```
          1)Chercher un meilleur partenaire
                – dans l'intersection des champs de perception et de
                      migration actuels
                – en évaluant les couples existants : un bon couple
                      ne doit pas être brisé

          2) Si un meilleur partenaire existe {
                2a) Envoyer message « PEUX EMIGRER ? » au super-agent
                2b) Envoyer message « PEUX IMMIGRER ? » au
                      destinataire
                2c) Attendre les réponses
                2d) Si les permissions sont accordées et si libre {
                      libre := faux
                      Envoyer message « EMIGRANT » au super-agent
                      Envoyer message « IMMIGRANT » au destinataire
                      Migrer
                            – en utilisant cet algorithme, le
                                  destinataire réserve une place pour
                                  l'immigrant ; celui n'a qu'à la
                                  prendre
                      libre := vrai
                } alors {
                      Envoyer message « NON EMIGRANT » au super-agent
                      Envoyer message « NON IMMIGRANT » au
                            destinataire
                }
          }

    }
}
```

***THREAD* « messages »**

```
{
    filsMigrant := faux ;
    Répéter chaque fois un message est reçu {

          si le message reçu == « PEUX EMIGRER ? » {
                si filsMigrant ou ! libre {
                      Répondre « NON »
                } alors {
                      Répondre « OUI »
                      filsMigrant := vrai
                }
          }

          si le message reçu == « PEUX IMMIGRER ? » {
                si filsMigrant ou ! libre {
                      Répondre « NON »
                alors {
                      Répondre « OUI »
                      libre := faux
                }
          }

          si le message reçu == « EMIGRANT » {
                Rendre sa place dans la structure au fils restant
                      - ceci peut s'effectuer de manières différentes
                S'auto-détruire
          }
```

```
            si le message reçu == « IMMIGRANT » {
                  Créer un agent composé
                  Rendre sa place dans la structure à l'agent composé
                        - ceci peut s'effectuer de manières différentes
                  Prendre la place d'un des fils de l'agent composé
                  Donner la place de l'autre fils à l'agent immigrant
                  libre := vrai
            }

            si le message reçu == « NON EMIGRANT » {
                  filsMigrant := faux
            }
            si le message reçu == « NON IMMIGRANT » {
                  libre := vrai
            }

            si le message reçu == « OUI » {
                  Notifier le thread « qui se ressemble, s'assemble »
            }

            si le message reçu == « NON » {
                  Notifier le thread « qui se ressemble, s'assemble »
            }

      }
  }
```

Notons que la variable `libre`, partagée par les deux *threads* doit être protégée elle-même contre des accès simultanés (par les deux *threads*). Heureusement, les langages modernes nous offrent les mécanismes nécessaires. En Java, par exemple, nous pouvons protéger la variable en utilisant l'instruction `synchronized`.

CHAPITRE V.1

EVALUATION ET PERSPECTIVES

The best way to predict the future is to invent it.
Alan Kay

V.1.1 A propos du modèle

Evaluation

Dans cette thèse, nous avons présenté le modèle des Systèmes Multi-Agents Minimaux : les SMAM. Nous avons étudié les SMA purs, étant complètement abstraits, et les SMAM augmentés, liés à un monde extérieur par l'intermédiaire de leurs attributs. Nous nous sommes concentrés sur le niveau conceptuel du modèle. Toutefois, nous n'avons pas négligé les niveaux des algorithmes et de l'implémentation.

Au niveau conceptuel, notre modèle est minimal. Ceci était un de nos souhaits principaux et nous croyons que ce choix a été utile. A cause de sa nature stricte et simple, nous avons pu analyser le modèle à un degré relativement élevé. Par exemple, nos confrères mathématiciens ont pu démontrer assez facilement que le problème de l'organisation optimale d'un SMAM augmenté de type FRIENDS est NP-difficile. Le modèle étant sans ambiguïtés, il s'est prêté facilement à la formalisation. Cette formalisation, à son tour, nous a aidé à développer et raffiner le modèle. Nous avons identifié plusieurs propriétés importantes et quelques types spéciaux de SMAM tels que les SMAM linéaires.

Nous avons opté pour un modèle universel dans le sens que les SMAM sont des entités abstraites et qu'ils sont identifiés par leur organisation, i.e. leur structure. Ceci nous permet, en théorie, d'appliquer le modèle dans des domaines scientifiques différents. Dans le cadre de cette thèse, nous avons effectué de prudentes excursions dans la Systémique et la Vie Artificielle. Il semble que notre modèle peut être utile dans ces disciplines, mais seule une véritable étude interdisciplinaire peut garantir cette utilité.

Dans le contexte plus ciblé des Systèmes Multi-Agents, notre modèle est abstrait dans le sens qu'il ne présuppose aucune architecture particulière. Il nous permet de mieux comprendre les phénomènes de la dynamique organisationnelle et de la migration. Par exemple, notre modèle exprime quantitativement qu'une hiérarchie n'est équilibrée que lorsque les agents hauts dans la hiérarchie sont d'une complexité (entropie) plus importante que les agents bas. Notre modèle offre un nouveau point de vue pour analyser l'organisation et son évolution. Il constitue un outil qui peut être utile dans l'étude du maintien de l'intégrité fonctionnelle. Dans l'avenir, il peut nous aider à analyser des populations d'agents sur l'Internet.

Par contre, la nature abstraite du modèle le sépare des modèles et des architectures existants. Cette distance peut rendre son application difficile. De plus, le modèle des SMAM est très spécifique dans le sens qu'il exige un groupement récursif binaire des agents. Bien que cette décomposition puisse être traduite en une autre, cette

caractéristique implique une certaine incompatibilité entre notre modèle et la plupart des modèles SMA existants. Elle fait à la fois la force et la faiblesse du modèle.

Dans cette thèse, le niveau algorithmique joue un rôle secondaire au niveau conceptuel. Toutefois, nous avons expérimenté un certain nombre d'algorithmes, centralisés et distribués. Nous n'avons pas essayé de trouver l'algorithme « parfait ». Par contre, nous avons essayé d'estimer la difficulté de la réorganisation optimale des SMAM purs et augmentés. Il s'avère qu'il s'agit d'un problème difficile. Dans les SMAM, un algorithme distribué doit faire face au fait que, à cause des actions des agents, l'environnement d'un agent donné change constamment. Il s'agit d'une version abstraite du problème classique de l'Intelligence Artificielle où un agent autonome se trouve confronté à un environnement incertain et dynamique dont l'agent ne perçoit qu'une partie. Cette constatation nous a motivé pour formuler le problème comme un *benchmark* pour des agents cognitifs.

En ce qui concerne les SMAM augmentés et spécifiquement ceux de type FRIENDS, le problème de la réorganisation optimale n'est pas évident non plus. En effet, il est NP-difficile, ce qui constitue un argument pour approcher le problème en utilisant de la technologie SMA. Etant réaliste, la recherche d'un algorithme générant la solution optimale n'a qu'un faible intérêt.

A un niveau plus bas, la programmation des applications nous a appris que la nature binaire et récursive du modèle facilite considérablement son implémentation sur un ordinateur numérique. Le modèle s'adapte facilement à cette architecture physique.

Perspectives

Le modèle des SMAM est le résultat d'une longue évolution dont la réalisation de cette thèse ne constitue que la partie la plus récente. Toutefois, à une échelle globale, il s'agit d'un modèle très récent et relativement peu développé.

Au niveau conceptuel, les perspectives, et les travaux à effectuer, sont considérables. L'étude théorique des SMAM doit être poursuivie. Nombre de propriétés intéressantes sont à formuler et à démontrer. Par exemple, il semble exister une relation relativement simple entre la représentation binaire de l'entropie d'un SMAM et le SMAM même. Est-ce correct ? Si c'est le cas, cette propriété peut être exploitée pour implémenter des SMAM d'une manière plus efficace ou même pour construire des architectures physiques spécialisées.

Le modèle des SMAM ne spécifie pas l'algorithme des agents. Il est possible qu'on puisse construire un algorithme « universel » permettant le codage des algorithmes comme des SMAM faisant partie des agents. L'évolution d'un tel SMAM implique l'évolution des algorithmes au sein des agents, rapprochant le modèle des algorithmes génétiques. La recherche d'un tel algorithme et de son codage associé nous semble très intéressante.

Dans un contexte SMA, l'étude de la relation entre les SMAM et les organisations classiques doit être approfondie. Spécifiquement, il nous semble utile d'étudier comment les concepts de groupe et de ressemblance peuvent être exprimés symboliquement et utilisés comme base d'un langage de communication entre agents hétérogènes. La construction d'un dictionnaire « concept — représentation SMAM » peut faire partie de ces travaux. Dans une telle représentation, des concepts comme « couple » et « migrant » sont probablement les plus faciles à exprimer, déterminant - à un certain degré - son champ d'utilisation. Le but ultime de ce projet est de trouver des solutions au problème de l'enracinement des symboles.

Au niveau des algorithmes, un très grand travail reste à effectuer. D'un côté, la recherche systématique d'algorithmes centralisés et distribués peut approfondir notre compréhension du problème. D'un autre côté, une solution naturelle au problème implique l'utilisation d'agents cognitifs, sujet d'étude de l'Intelligence Artificielle classique. Dans ce contexte, la conception et l'implémentation d'agents adaptés, et leur comparaison à l'aide d'un *benchmark* (comme celui proposé dans cette thèse) constitue un projet très intéressant.

En dehors du domaine de l'Intelligence Artificielle, nous pouvons également identifier des perspectives intéressantes. Dans la Vie Artificielle, nous croyons que le développement de systèmes autopoiétiques au sein d'un SMAM peut mener à une meilleure compréhension du sujet de la discipline. Il s'agit d'un problème difficile, mais fascinant.

Lorsqu'on arrive à coder des algorithmes comme des SMAM, ces SMAM seront comparables aux *memes* de l'étude de l'évolution des idées. Dans ce cas, l'évolution naturelle d'un SMAM correspond à l'évolution de cultures différentes vers une culture universelle. Vu les origines philosophiques de nos travaux, les perspectives d'une telle simulation nous fascinent. Nous nous rendons compte qu'il s'agit d'une perspective à long terme, impliquant des travaux interdisciplinaires considérables.

Encore plus éloignée de l'Intelligence Artificielle se trouve la Physique. Nous croyons que les SMAM, comme les SMA en général, peuvent jouer un rôle dans la modélisation physique. Une coopération interdisciplinaire pourrait valider cette intuition.

Seules des recherches interdisciplinaires peuvent vérifier si notre modèle est vraiment d'une nature multidisciplinaire. Toutefois, sa nature minimale joue en sa faveur. Peut-être le modèle est un *meme* atomique, présent - implicitement ou explicitement - dans un grand nombre de disciplines. L'étude de cette hypothèse nous semble très intéressante.

V.1.2 A propos des applications

Evaluation

La première application - dans le sens informatique du mot - que nous avons développé est FRIENDS Offline. Malgré son nom, ce programme est une plate-forme générale pour expérimenter les SMAM purs et un type de SMAM augmentés. Nous avons utilisé intensivement cette plate-forme dans le développement de notre modèle. Inversement, le modèle théorique, dans ses phases différentes, a guidé l'évolution de notre plate-forme. Cette expérience a affirmé notre opinion que les outils interactifs et multimédias peuvent être très utiles dans l'étude des systèmes complexes.

Les deux autres applications que nous avons construites sont toutes les deux basées sur le concept de FRIENDS. La première, FRIENDS Numbercruncher, a été conçue et optimisée pour le traitement de données statiques. En particulier, elle est configurée pour traiter des données compilées au sein de France Télécom.

Au niveau technique, le fonctionnement de l'application est correct. Dans tous nos essais, nous n'avons rencontré aucun problème. Au niveau applicatif, des chercheurs de France Télécom ont effectué un certain nombre de tests. Les résultats de ces tests sont très positifs. En particulier, le test le plus important, effectué sur une population de presque 5000 agents et impliquant plus de 146000 mots-clés, illustre le potentiel de l'application. Il a permis l'identification automatique de catégories de classification importantes.

Notons qu'il ne s'agit que d'une première série d'essais et que les résultats ne sont pas encore complètement analysés. Afin de faciliter cette analyse, l'interface homme-machine de l'application doit être améliorée. Par exemple, une interface graphique, plutôt qu'une interface textuelle serait plus adaptée. Notons également qu'il existe certaines différences entre cette application et le concept original de FRIENDS, ce qui limite sa fonctionnalité. Par exemple, dans la version statique, la normalisation des mots-clés peut être difficile, puisque leur choix n'est pas guidé.

Notre application principale est FRIENDS Online. Elle existe en deux versions : FRIENDS Online 3.1 et FRIENDS Online MAI. La deuxième version, utilisant la technologie des agents mobiles, est expérimentale et doit être située dans une étude de faisabilité. Par contre, la première version, du type client - serveur, est d'une grande stabilité et adaptée à une utilisation en ligne. Elle est conçue pour des expériences de longue durée impliquant jusqu'à 500 utilisateurs.

Nous l'avons testée à l'occasion de la conférence ICMAS'98 à Paris. Pendant une semaine, le système, composé d'un serveur à Grenoble et d'un client à Paris, a fonctionné sans le moindre problème. Par contre, pour des raisons pratiques, seulement 50 personnes ont participé à l'expérience et son interactivité était limitée. Toutefois, l'expérience nous a été utile et elle a validé le côté technique du système. Pour mieux valider le concept, nous devons tester le système à une échelle plus importante. FRIENDS Online est conçu pour le Web, mais il nécessite des navigateurs récents supportant le JDK 1.1. Jusqu'à maintenant, ceci a limité l'accessibilité du service. Dans un avenir proche, cette contrainte technique ne posera plus un problème et un essai plus important pourra être effectué.

Perspectives

La plate-forme « FRIENDS Offline », bien que stable et utile, est loin d'être finie. Puisqu'elle suit l'évolution du modèle, elle ne sera finalisée que lorsque le modèle sera finalisé. Considérant nos perspectives pour le modèle, sa finalisation dans un avenir proche est peu probable. Dans un premier temps, nous souhaitons finaliser les fonctions de base, telles que l'aide en ligne, et ajouter des méthodes de visualisation encore plus puissantes et intuitives.

En ce qui concerne « FRIENDS Numbercruncher », nous croyons qu'il existe toute une gamme de domaines où cette application et des applications similaires peuvent être utiles. Par exemple, le type de regroupement inspiré par notre modèle et effectuée par FRIENDS Numbercruncher peut être exploité pour construire des taxonomies, des classifications morphologiques ou même certains types d'ontologies.

Notons que, pour des populations à partir d'une certaine taille, FRIENDS Numbercrucher demande trop de temps de calcul pour être pratique. Dans ce cas, le développement de nouveaux algorithmes et/ou de versions distribuées est indispensable. La recherche sur des SMAM composés d'agents cognitifs (cf. notre *benchmark*) peut contribuer à ce développement.

FRIENDS Online est notre application principale. Toutefois, nous n'avons implémenté que la base du système. Pour exploiter pleinement son potentiel, nous devons améliorer son interface homme - machine et ajouter une couche fonctionnelle supérieure. En particulier, la décomposition binaire doit être cachée à l'utilisateur lorsqu'elle n'est pas pertinente pour lui. La couche supérieure peut offrir des services permettant aux utilisateurs de partager des informations ou de mener des conversations dans des groupes de taille variable.

Nous croyons que l'intérêt de FRIENDS Online croît avec la taille de la population d'utilisateurs. Un essai du système à une grande échelle (impliquant des dizaines de milliers d'utilisateurs (ou plus) et sur une période d'au moins une année) nous semble très intéressant. Evidemment, un tel essai nécessite un modèle commercial pour générer les ressources nécessaires. De plus, il exige une implémentation efficace et probablement distribuée du système. Dans ce contexte aussi, des travaux théoriques sur les agents cognitifs peuvent s'avérer utiles.

En dehors des applications existantes, d'autres systèmes basés sur le modèle des SMAM peuvent être conçus. Une première application possible combine les concepts de FRIENDS Numbercruncher et de FRIENDS Online. FRIENDS Numbercruncher, comme il est utilisé par France Télécom, groupe des services. FRIENDS Online groupe des utilisateurs. Nous pouvons construire un système mélangeant les deux types d'éléments. Dans un tel système, les utilisateurs se trouveront près de services correspondant à leurs intérêts.

Dans un contexte robotique, nous envisageons l'application de notre modèle à la classification sensorielle. Par exemple, le modèle de Luc Steels, décrit dans [Steels 96b], semble être un candidat convenable pour son intégration. Dans un domaine très différent, nous souhaitons explorer la création musicale à l'aide des SMAM. Il s'agit d'un problème fascinant et peu compris. De plus, il est directement lié à l'Intelligence Artificielle et il implique, malgré sa nature mentale, le monde physique. Il n'est pas garanti que notre outil simple puisse nous aider à mieux comprendre le phénomène, mais nous souhaitons l'essayer.

V.1.3 A propos de la thèse

Au début de ce mémoire, nous avons présenté la thèse suivante :

— Un modèle minimal des Systèmes Multi-Agents peut être basé sur le concept du couple, et le principe « qui se ressemble, s'assemble ».

— Un tel modèle peut mener à une meilleure compréhension de la dynamique organisationnelle dans les Systèmes Multi-Agents.

— Un tel modèle peut être appliqué à des problèmes importants.

Est-ce que nous avons, avec succès, défendu cette thèse ? Répondons à cette question point par point :

— Nous avons développé un modèle minimal basé sur le concept du couple, et le principe « qui se ressemble, s'assemble ». Nous avons nommés les systèmes décrits par ce modèle « les Systèmes Multi-Agents Minimaux » (SMAM). Il s'agit d'un modèle très simple modélisant une classe de Systèmes Multi-Agents (SMA) à la fois très générique et très spécifique. D'un côté, les SMAM ont des caractéristiques communes de tous les SMA. D'un autre côté, leur structure et leur comportement sont tellement spécifiques que peu d'autres types de SMA peuvent directement être modélisés par le modèle.

— La nature de notre modèle a permis une étude quantitative de la dynamique organisationnelle dans les SMAM. Nous avons pu formaliser et quantifier quelques intuitions générales sur la dynamique organisationnelle et sur la migration.

— A partir de notre modèle, nous avons construit plusieurs applications apportant des nouvelles solutions à des problèmes importants. Des essais montrent que le système FRIENDS, inspiré directement par le modèle, a un potentiel réel.

ANNEXE I

UN PROBLEME NP-DIFFICILE

Sur le problème de l'organisation optimale d'un Système Multi-Agents Minimal augmenté de type FRIENDS

Frédéric Maffray et Myriam Preissmann
Equipe Graphes, Laboratoire Leibniz - IMAG
46, avenue Félix Viallet, 38031 Grenoble Cedex

Ce problème nous a été posé par Daniel Genovese et Francis Van Aeken de l'équipe MAGMA de Leibniz - IMAG. Nous l'exprimons ici dans le langage des Maths Discrètes. Soit $\mathcal{F} = \{S_1, \ldots, S_n\}$ une famille d'ensembles finis. Nous appelons *arbre binaire d'intersections* (ABI) de $\mathcal{F}$ n'importe quel arbre binaire (où chaque nœud intérieur a exactement deux enfants) A muni d'une correspondance $S : N(A) \to 2^{S_1 \cup \ldots \cup S_n}$ (où $N(A)$ est l'ensemble des nœuds de A) ayant les propriétés suivantes :

- A a exactement n feuilles $a_1, \ldots, a_n$;
- Pour chaque feuille a_i on a $S(a_i) = S_i$;
- Pour chaque nœud intérieur a de A on a $S(a) = S(y) \cap S(z)$, où y, z sont les deux enfants de a dans A.

Etant donné un arbre binaire d'intersections A d'une famille $\mathcal{F}$, la *valeur* de chaque nœud a de A est $v(a) = |S(a)|$. La valeur de l'arbre est $v(A) = \sum\{v(a) \mid a \text{ nœud intérieur de } A\}$. Une famille $\mathcal{F}$ peut admettre plusieurs ABI, éventuellement de valeurs différentes. Notons $v(\mathcal{F}) = \max \{v(A) \mid A \text{ est un ABI de } \mathcal{F}\}$ et appelons ABI *optimal* tout ABI A tel que $v(A) = v(F)$.

Problème ABI-max : *Etant donnée une famille $\mathcal{F}$, déterminer la valeur de $v(\mathcal{F})$ et un ABI optimal.*

Théorème 1 *Le problème ABI-max est NP-difficile, même lorsqu'il est restreint à la classe des instances $\mathcal{F}$ dont les éléments sont de taille 2 et tous différents.*

Avant de donner la preuve de ce théorème, faisons quelques remarques. Nous dirons que $\mathcal{F} = \{S_1, \ldots, S_n\}$ est une *famille étoilée* de centre S_0 ($S_0 \subseteq S_1 \cup \ldots \cup S_n$) si pour toute paire $i, j \in \{1, \ldots, n\}$ ($i \neq j$) on a $S_i \cap S_j = S_0$. Notons que tous les ABI ayant n feuilles ont exactement n - 1 nœuds intérieurs ; ceci implique le lemme suivant.

Lemme 1 *Si $\mathcal{F}$ est une famille étoilée de centre S_0, tout ABI de $\mathcal{F}$ est de valeur* $(n - 1)|S_0|$. □

Dans un arbre, le sous-arbre formé par un nœud et tous ses successeurs sera appelé un *sous-arbre terminal*.

Dans un graphe simple $G = (V, E)$, une *étoile* est un ensemble d'arêtes incidentes à un sommet commun, appelé *centre* de l'étoile. Un *transversal* est un ensemble $T \subseteq V$ tel que chaque arête de G a au moins une extrémité dans T. Soit $\tau(G)$ la taille minimum d'un transversal. Déterminer $\tau(G)$ et trouver un transversal de taille $\tau(G)$ est un

problème NP-difficile [1]. Il est facile de vérifier que $\tau(G)$ est égal à la taille minimum d'une partition des arêtes de G en étoiles.

Dans le lemme suivant, étant donné un graphe simple $G = (V, E)$ nous considérons l'instance du problème ABI-max donné par $\mathcal{F} = E$.

Lemme 2 *Soit $G = (V, E)$ un graphe simple. Alors $\nu(E) = |E| - \tau(G)$.*

Démonstration du lemme 2. On vérifie facilement que dans tout ABI de $\mathcal{F}$ la valeur d'un nœud intérieur est 0 ou 1. Nous affirmons que :

> Pour toute partition P des arêtes de G en étoiles, il existe un ABI de $\mathcal{F}$, noté $A(P)$, dont la valeur satisfait $\nu(A(P)) \geq |E| - |P|$,

et que

> Pour tout ABI A de $\mathcal{F}$ (= E), il existe une partition P des arêtes de G en étoiles telle que $|P| = |E| - \nu(A)$.

Les deux affirmations (1) et (2) impliquent immédiatement le lemme.

Pour prouver (1), posons $P = \{E_1, \ldots, E_p\}$. Pour $i = 1, \ldots, p$, soit A_i un ABI quelconque de E_i ; comme E_i est une famille étoilée, cet arbre est de valeur $\nu(A_i) = |E_i| - 1$ par le lemme 1. Maintenant, soit $A(P)$ un ABI dont les feuilles sont les éléments de E et tel que les A_i soient des sous-arbres terminaux de $A(P)$. Il suit que

$$\nu(A(P)) \geq \sum_{i=1}^{p} \nu(A_i) = \sum_{i=1}^{p} (|E_i| - 1) = \sum_{i=1}^{p} |E_i| - r = |E| - |P| \ . \square$$

(L'inégalité ci-dessus est due au fait que deux étoiles peuvent avoir le même centre.)

Pour prouver (2), soit A un ABI quelconque de $\mathcal{F}$. Soit U l'ensemble des nœuds u de A tels que $\nu(u) > 0$ et $\nu(u') = 0$, où u' est le parent de u dans A (si la racine r de A satisfait $\nu(r) > 0$ alors $U = \{r\}$). Pour chaque $u \in U$, soit A_u le sous-arbre terminal de A de racine u, et soit E_u l'ensemble des arêtes de G qui sont les feuilles de A_u. On remarque que E_u est une étoile car $\nu(u) > 0$, et que $P = \{E_u \mid u \in U\}$ est une partition de E, avec $|P| = |U|$. De plus,

$$\nu(A) = \sum_{u \in U} \nu(A_u) = \sum_{u \in U} (|E_u| - 1) = |E| - |P| \ . \square$$

Démonstration du théorème. Le lemme 2 montre que le problème du transversal minimum dans un graphe $G = (V, E)$ est équivalent au problème ABI-max pour $\mathcal{F} = E$. Ceci montre bien que le problème ABI-max est NP-difficile même lorsqu'il est restreint à la classe des instances sont tous les éléments sont de taille 2 et différents. □

Références

[1] M.R. Garey et D.S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

ANNEXE II

CLASSIFICATION DE SERVICES

Dans cette annexe, nous présentons l'arborescence - jusqu'au niveau 7 - du SMAM produit par FRIENDS Numbercruncher dans l'expérience # 3 effectuée par les chercheurs de France Télécom. L'expérience impliquait 4997 services, décrits par 146674 mots-clés dont 16384 différents. FRIENDS Numbercruncher a construit le SMAM complet en 70337 secondes (19 heures et demie). La plate-forme est inconnue.

(1)
(1.1)
(1.1.1)
(1.1.1.1)
(1.1.1.1.1)
(1.1.1.1.1.1) religion
(1.1.1.1.1.1.1) religion
(1.1.1.1.1.1.2) religion
(1.1.1.1.2) linguistique
(1.1.1.1.2.1) linguistique
(1.1.1.1.2.2) linguistique
(1.1.1.2)
(1.1.1.2.1) SPORT
(1.1.1.2.1.1) SPORT
(1.1.1.2.1.2) SPORT
(1.1.1.2.2) POLITIQUE
(1.1.1.2.2.1) POLITIQUE
(1.1.1.2.2.2) POLITIQUE
(1.1.2)
(1.1.2.1)
(1.1.2.1.1) alimentation
(1.1.2.1.1.1) alimentation
(1.1.2.1.1.2) alimentation
(1.1.2.1.2) ZOOLOGIE
(1.1.2.1.2.1) ZOOLOGIE
(1.1.2.1.2.2) ZOOLOGIE
(1.1.2.2)
(1.1.2.2.1) LITTERATURE
(1.1.2.2.1.1) LITTERATURE
(1.1.2.2.1.2) LITTERATURE
(1.1.2.2.2) musique
(1.1.2.2.2.1) musique
(1.1.2.2.2.2) musique
(1.1.2)
(1.1.2.1)
(1.1.2.1.1)
(1.1.2.1.1.1) commerce
(1.1.2.1.1.1.1) commerce
(1.1.2.1.1.1.2) commerce
(1.1.2.1.1.2) media
(1.1.2.1.1.2.1) media
(1.1.2.1.1.2.2) media
(1.1.2.1.2)
(1.1.2.1.2.1) droit
(1.1.2.1.2.1.1) droit
(1.1.2.1.2.1.2) droit
(1.1.2.1.2.2) medecine
(1.1.2.1.2.2.1) medecine
(1.1.2.1.2.2.2) medecine
(1.1.2.2)
(1.1.2.2.1)
(1.1.2.2.1.1) art
(1.1.2.2.1.1.1) art
(1.1.2.2.1.1.2) art
(1.1.2.2.1.2) transports
(1.1.2.2.1.2.1) transports
(1.1.2.2.1.2.2) transports
(1.1.2.2.2)
(1.1.2.2.2.1) EDUCATION
(1.1.2.2.2.1.1) EDUCATION
(1.1.2.2.2.1.2) EDUCATION
(1.1.2.2.2.2) telecom
(1.1.2.2.2.2.1) telecom
(1.1.2.2.2.2.2) telecom
(1.2)
(1.2.1)
(1.2.1.1)
(1.2.1.1.1)
(1.2.1.1.1.1) informatique
(1.2.1.1.1.1.1) informatique
(1.2.1.1.1.1.2) informatique
(1.2.1.1.1.2) GEOGRAPHIE
(1.2.1.1.1.2.1) GEOGRAPHIE
(1.2.1.1.1.2.2) GEOGRAPHIE
(1.2.1.1.2)
(1.2.1.1.2.1)
(1.2.1.1.2.2)
(1.2.1.2)
(1.2.1.2.1)
(1.2.1.2.1.1)
(1.2.1.2.1.2)
(1.2.1.2.2)
(1.2.1.2.2.1)
(1.2.1.2.2.2)
(1.2.1.1.2.1.1) laPoste philatélie philatélique philatéliste
(1.2.2)
(1.2.2.1)
(1.2.2.1.1)
(1.2.2.1.1.1)
(1.2.2.1.1.1.1) canot kayak marine
(1.2.2.1.1.1.2) lunetier lunetterie optique
(1.2.2.1.1.2)
(1.2.2.1.1.1.1) comptabilite comptable

	compte
(1.2.2.1.1.1.2)	fumer tabac
(1.2.2.1.2)	
(1.2.2.1.2.1)	
(1.2.2.1.2.1.1)	boucherie charcuterie
(1.2.2.1.2.1.2)	décoration tissu
(1.2.2.1.2.2)	
(1.2.2.1.2.2.1)	model Top
(1.2.2.1.2.2.2)	Chagall Picasso
(1.2.2.2)	
(1.2.2.2.1)	
(1.2.2.2.1.1)	
(1.2.2.2.1.1.1)	FM radio
(1.2.2.2.1.1.2)	ANATOMIE sein
(1.2.2.2.1.2)	
(1.2.2.2.1.2.1)	ecologie écologie
(1.2.2.2.1.2.2)	Elevage equitation
(1.2.2.2.2)	
(1.2.2.2.2.1)	
(1.2.2.2.2.1.1)	danse danser
(1.2.2.2.2.1.2)	automobile voiturer
(1.2.2.2.2.2)	
(1.2.2.2.2.2.1)	archeologie archéologique
(1.2.2.2.2.2.2)	ASTRONOMIE planète
(2)	
(2.1)	
(2.1.1)	
(2.1.1.1)	
(2.1.1.1.1)	
(2.1.1.1.1.1)	Ecole école
(2.1.1.1.1.2)	PHOTO photographie
(2.1.1.1.2)	
(2.1.1.1.2.1)	pièce theatre
(2.1.1.1.2.2)	meteorologie météorologie
(2.1.1.2)	
(2.1.1.2.1)	
(2.1.1.2.1.1)	cinema film
(2.1.1.2.1.2)	science scientifique
(2.1.1.2.2)	
(2.1.1.2.2.1)	science scientifique
(2.1.1.2.2.2)	animateur
(2.1.2)	
(2.1.2.1)	
(2.1.2.1.1)	
(2.1.2.1.1.1)	production
(2.1.2.1.1.2)	vendeur
(2.1.2.1.2)	
(2.1.2.1.2.1)	social
(2.1.2.1.2.2)	international
(2.1.2.2)	
(2.1.2.2.1)	
(2.1.2.2.1.1)	écrit
(2.1.2.2.1.2)	direct
(2.1.2.2.2)	
(2.1.2.2.2.1)	nationale
(2.1.2.2.2.2)	financier
(2.1.2)	
(2.1.2.1)	
(2.1.2.1.1)	
(2.1.2.1.1.1)	
(2.1.2.1.1.1.1)	collection
(2.1.2.1.1.1.2)	packaging
(2.1.2.1.1.2)	
(2.1.2.1.1.2.1)	vente
(2.1.2.1.1.2.2)	bordeaux
(2.1.2.1.2)	
(2.1.2.1.2.1)	
(2.1.2.1.2.1.1)	truc
(2.1.2.1.2.1.2)	mer
(2.1.2.1.2.2)	
(2.1.2.1.2.2.1)	réseau
(2.1.2.1.2.2.2)	elements
(2.1.2.2)	
(2.1.2.2.1)	
(2.1.2.2.1.1)	
(2.1.2.2.1.1.1)	Cosmétiques
(2.1.2.2.1.1.2)	bateau
(2.1.2.2.1.2)	
(2.1.2.2.1.2.1)	multimédia
(2.1.2.2.1.2.2)	laboratoire
(2.1.2.2.2)	
(2.1.2.2.2.1)	
(2.1.2.2.2.1.1)	metallurgie
(2.1.2.2.2.1.2)	aviation
(2.1.2.2.2.2)	
(2.1.2.2.2.2.1)	ouvrage
(2.1.2.2.2.2.2)	CHIMIE
(2.2)	
(2.2.1)	
(2.2.1.1)	
(2.2.1.1.1)	
(2.2.1.1.1.1)	
(2.2.1.1.1.1.1)	AGRICULTURE
(2.2.1.1.1.1.2)	SCIENCESSOCIALES
(2.2.1.1.2)	
(2.2.1.1.2.1)	terre
(2.2.1.1.2.2)	construction
(2.2.1.2)	
(2.2.1.2.1)	
(2.2.1.2.1.1)	biologie
(2.2.1.2.1.2)	famille
(2.2.1.2.2)	
(2.2.1.2.2.1)	occultisme
(2.2.1.2.2.2)	militaire
(2.2.1.2)	
(2.2.1.2.1)	
(2.2.1.2.1.1)	
(2.2.1.2.1.1.1)	spectacle
(2.2.1.2.1.1.2)	BOTANIQUE
(2.2.1.2.1.2)	
(2.2.1.2.1.2.1)	ARGENT
(2.2.1.2.1.2.2)	création
(2.2.1.2.2)	
(2.2.1.2.2.1)	
(2.2.1.2.2.1.1)	en-ligne
(2.2.1.2.2.1.2)	annuaire
(2.2.1.2.2.2)	
(2.2.1.2.2.2.1)	économique
(2.2.1.2.2.2.2)	histoire
(2.2.2)	
(2.2.2.1)	
(2.2.2.1.1)	
(2.2.2.1.1.1)	
(2.2.2.1.1.1.1)	architecture
(2.2.2.1.1.1.2)	canada
(2.2.2.1.1.2)	
(2.2.2.1.1.2.1)	matériel
(2.2.2.1.1.2.2)	mécanique
(2.2.2.1.2)	
(2.2.2.1.2.1)	
(2.2.2.1.2.1.1)	mobilier
(2.2.2.1.2.1.2)	web
(2.2.2.1.2.2)	

(2.2.2.1.2.2.1) edition
(2.2.2.1.2.2.2) ARTISANAT
(2.2.2.2)
(2.2.2.2.1)
(2.2.2.2.1.1)
(2.2.2.2.1.1.1) actualité
(2.2.2.2.1.1.2) gestion
(2.2.2.2.1.2)
(2.2.2.2.1.2.1) industrie
(2.2.2.2.1.2.2) ECONOMIE
(2.2.2.2.2)
(2.2.2.2.2.1)
(2.2.2.2.2.1.1) urbanisme
(2.2.2.2.2.1.2) TEXTILE
(2.2.2.2.2.2)
(2.2.2.2.2.2.1) ELECTRICITE
(2.2.2.2.2.2.2) boisson

ANNEXE III

FRIENDS OFFLINE

1. Introduction

Cette annexe constitue un manuel compact de FRIENDS Offline 1.7. Le logiciel nécessite Windows NT 4.0, Windows 95 ou Windows 98. La seule langue actuellement supportée est l'anglais. La taille de la version autonome (*Statically Linked*) du code exécutable est 393 Ko.

L'application ne nécessite pas de procédure d'installation et peut directement être lancée en double-cliquant sur son icône, montré dans la figure 1.



Figure 1 : L'icône de FRIENDS Offline 1.7.

La figure 2 montre une vue typique de FRIENDS Offline. Nous pouvons distinguer la barre de menus composée des éléments « Agent », « Edit », « Dynamics », « View », « Window », « Numbers » et « Help ». Au-dessous de la barre de menus se trouve la barre d'outils composée d'un nombre de boutons affichant des icônes. Les commandes principales des menus sont dupliquées dans cette barre. Tout en bas de la fenêtre principale se trouve la barre d'état indiquant des informations utiles. Entre la barre d'outils et la barre d'état se trouve la région principale dans laquelle sont affichées les fenêtres associées aux documents. Chaque document représente un SMAM. Notons que plusieurs documents peuvent être ouverts en même temps. En effet, FRIENDS Offline est une application MDI (*Multiple Document Interface*, comme Microsoft Word), non pas une application SDI (*Single Document Interface*, comme Microsoft Wordpad).

FRIENDS Offline permet aux utilisateurs de **créer** des SMAM (de types différents), de les **éditer**, de les **faire évoluer** et de les **afficher** de manières différentes. De plus, des représentations graphiques des SMAM et de leur évolution peuvent être **exportées** (copier - coller) et **imprimées**.

L'application est préparée pour être équipée des fonctions de sauvegarde et d'aide. Elles ne sont pas implémentées dans la version 1.7, mais elle sont prévues pour la version 1.8.

En plus de la fenêtre principale, des fenêtres secondaires peuvent être créées par l'application, typiquement pour afficher des représentations graphiques alternatives ou pour afficher l'évolution de l'entropie.

Parcourons systématiquement les fonctionnalités de l'application. Dans ce qui suit, l'expression « A / B » signifie la commande « B » du menu « A ». Notons que la plupart des commandes des menus peuvent être accédées par la barre d'outils. Il suffit de promener le pointeur au-dessus des boutons pour activer des « tool tips » identifiant leur fonction.

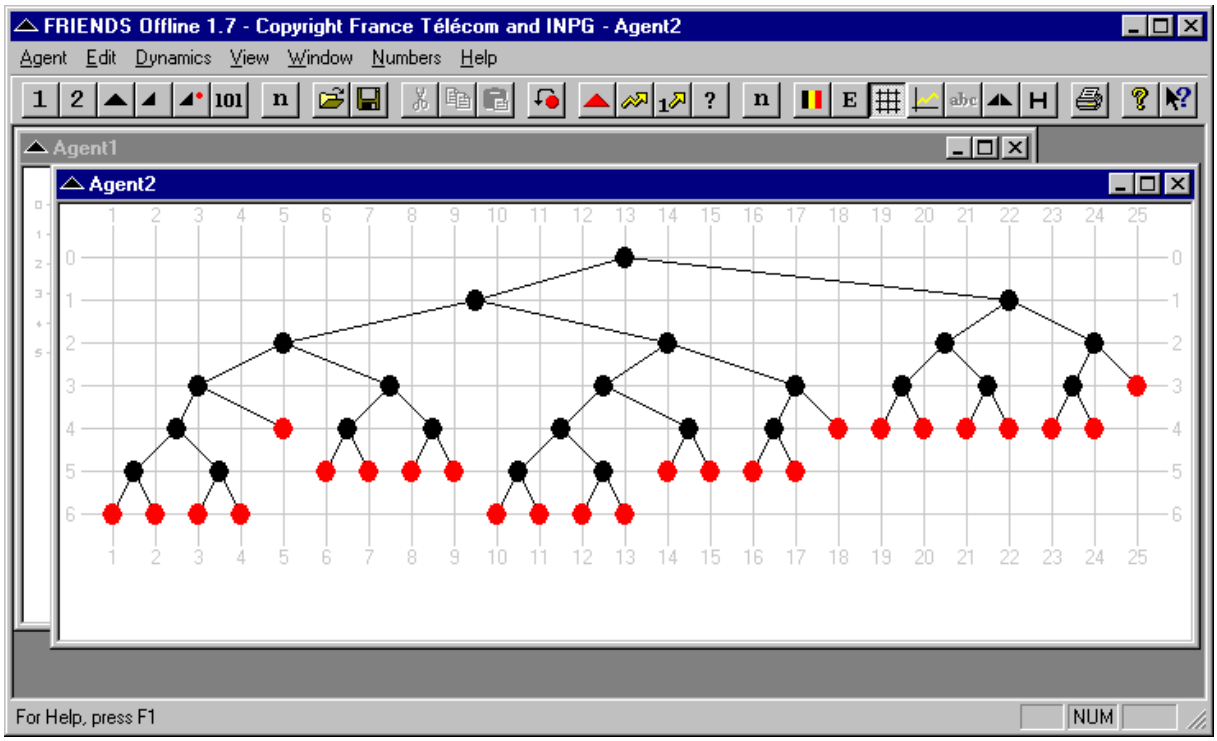


Figure 2 : Fenêtre principale de FRIENDS Offline 1.7.

2. Créer des SMAM

Lorsque le programme est lancé, un SMAM augmenté, et donc un document, est automatiquement créé. Comme tout document, il peut être fermé en appuyant sur le bouton « X » en haut à droite de la fenêtre. La sélection de la commande « Agent / Close » a le même effet.

Les commandes du menu « Agent » permettent la création de six types différents de SMAM. Il s'agit des commandes « Agent / Unary », « Agent / Binary », « Agent / Fair », « Agent / Proportional », « Agent / Odd » et « Agent / Attributed ».

La commande « Agent / Unary »

La sélection de cette commande crée un SMAM linéaire. La taille réelle du SMAM fait partie des paramètres de création. Ces paramètres peuvent être modifiés en choisissant « Agent / Parameters... ». La sélection de cette commande affiche la fenêtre « Parameters » de la figure 3. Par l'intermédiaire de cette fenêtre, la taille réelle des nouveaux SMAM peut être modifiée. Dans la fenêtre, le champ « # Agents » réfère au nombre d'agents *atomiques*.

Notons que la représentation initiale utilisée par FRIENDS Offline est la représentation arborescente *compacte*. Elle peut être changée en non compacte, et vice versa, en sélectionnant « Dynamics / Aggregate ».

La commande « Agent / Binary »

La sélection de cette commande crée un SMAM dont la structure correspond à la représentation binaire de la taille réelle. La taille réelle est celle spécifiée dans la fenêtre « Parameters ».

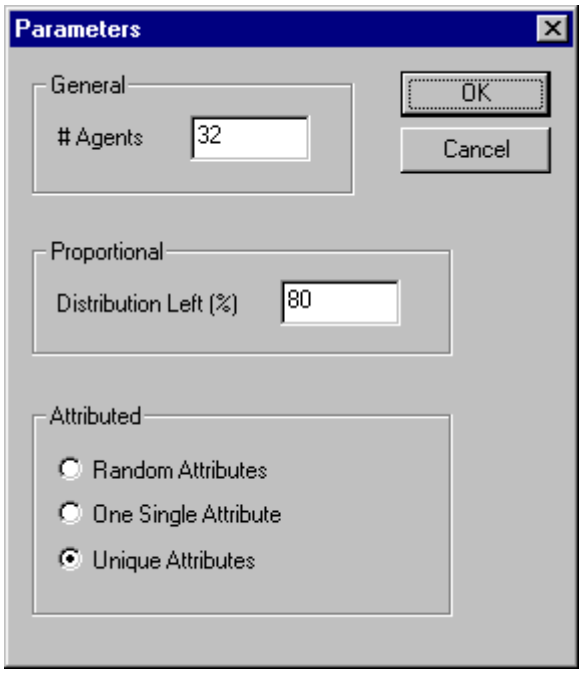


Figure 3 : La fenêtre « Parameters ».

La commande « Agent / Fair »

La sélection de cette commande crée un SMAM d'un équilibre maximale pour une taille donnée. La taille réelle est celle spécifie dans la fenêtre « Parameters ».

La commande « Agent / Proportional »

La sélection de cette commande crée un SMAM dans lequel la proportion des tailles réelles des deux fils de chaque agent est déterminé par le paramètre « Distribution Left » de la fenêtre « Parameters ». La taille réelle du SMAM est celle spécifiée dans la même fenêtre.

La commande « Agent / Odd »

La sélection de cette commande crée un SMAM dans lequel les tailles réelles des deux fils de chaque agent sont identiques, sauf si leur somme est impaire. Dans ce cas, un des deux fils est un agent atomique. La taille réelle du SMAM total est celle spécifie dans la fenêtre « Parameters ».

La commande « Agent / Attributed »

La sélection de cette commande crée un SMAM augmenté dont chaque agent a comme attribut un vecteur binaire de taille 5. Le choix des valeurs des attributs est déterminé par les paramètres de création. La fenêtre « Parameters » offre trois options. Lorsque l'option « Random Attributes » est choisi, des valeurs aléatoires sont attribuées. La sélection de l'option « One Single Attribute » attribue la même valeur à chaque agent. Si l'option « Unique Attributes » est choisi, tous les agents ont des attributs différents (si cela est possible). La taille réelle est celle spécifie dans la fenêtre « Parameters »

Lorsque l'on promène le pointeur au-dessus d'un agent attribué, une fenêtre affichant une interprétation de son attribut est créé. Un exemple d'une telle fenêtre est montré dans la figure 4, détaillant une partie d'un document de FRIENDS Offline. Cette fonction peut être activée ou désactivée en sélectionnant « View / Keywords ».

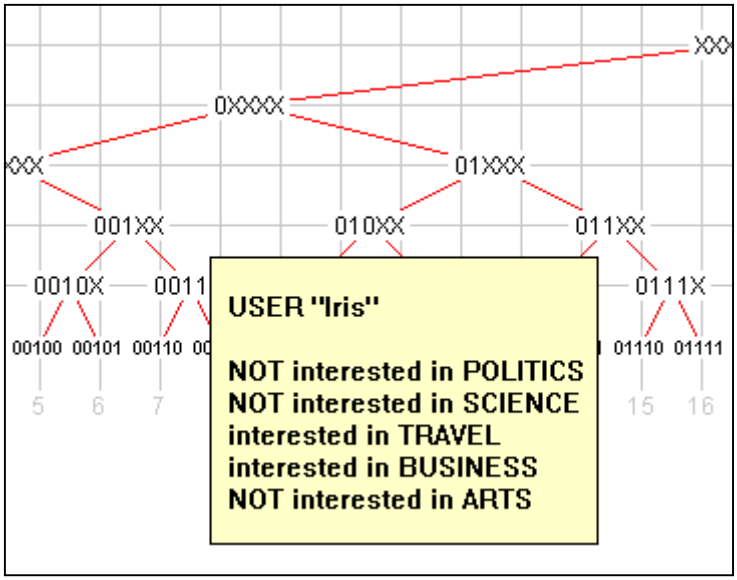


Figure 4 : Interprétation automatique d'un attribut.

3. Editer des SMAM

La manipulation la plus simple est l'ajout d'un agent atomique au SMAM actif (le SMAM dans la fenêtre active). Cette opération est effectuée en sélectionnant « Edit / Add Agent ». Des manipulations plus complexes nécessitent généralement la sélection d'un ou plusieurs agents.

Un agent d'un SMAM peut être sélectionné en cliquant dessus. Ceci entraîne la sélection de tous les sous-agents. En effet, un agent composé représente le SMAM complet composé de tous les sous-agents. Les agents sélectionnés sont affichés de couleurs différentes. Comme nous le montrons plus tard, ces couleurs peuvent être personnalisées.

Une fois qu'un agent est sélectionné, il peut être manipulé. Sa plus simple manipulation est sa destruction. Elle est effectuée en sélectionnant « Edit / Cut (split) ». Cette opération copie l'agent dans le presse-papiers interne. Il peut également être copié, sans le détruire, en sélectionnant « Edit / Copy ».

Une fois un agent copié dans le presse-papiers interne, il peut être joint à un autre agent. Il suffit de sélectionner le destinataire, puis de sélectionner « Edit / Paste (join) ».

Si on souhaite faire migrer un agent dans le SMAM, il n'est pas nécessaire de passer par le presse-papiers. Il suffit de le faire glisser vers le destinataire. Pendant ce processus, le pointeur change de forme, et le destinateur sélectionné change de couleur. Le destinataire potentiel constitue une sélection *secondaire* dont la couleur peut également être personnalisée.

Des actions peuvent être annulées en sélectionnant « Edit / Undo ». La mémoire contenant les versions antérieures du SMAM peut être libérée en sélectionnant « Edit / Reset History ». Une fois la mémoire libérée, les opérations effectuées auparavant ne peuvent plus être annulées.

Notons que les commandes « Undo », « Cut », « Copy » et « Paste » peuvent être exécutées en utilisant respectivement les combinaisons de touches « Ctrl+Z », « Ctrl+X », « Ctrl+C » et « Ctrl+V ».

4. Faire évoluer des SMAM

FRIENDS Offline permet l'expérimentation de l'évolution des SMAM. La sélection de « Dynamics / Run », fait évoluer le SMAM actif (dont la fenêtre est sélectionnée) selon les paramètres choisis. La sélection de « Dynamics / Parameters... » affiche la fenêtre de la figure 5, permettant le choix des paramètres d'évolution.

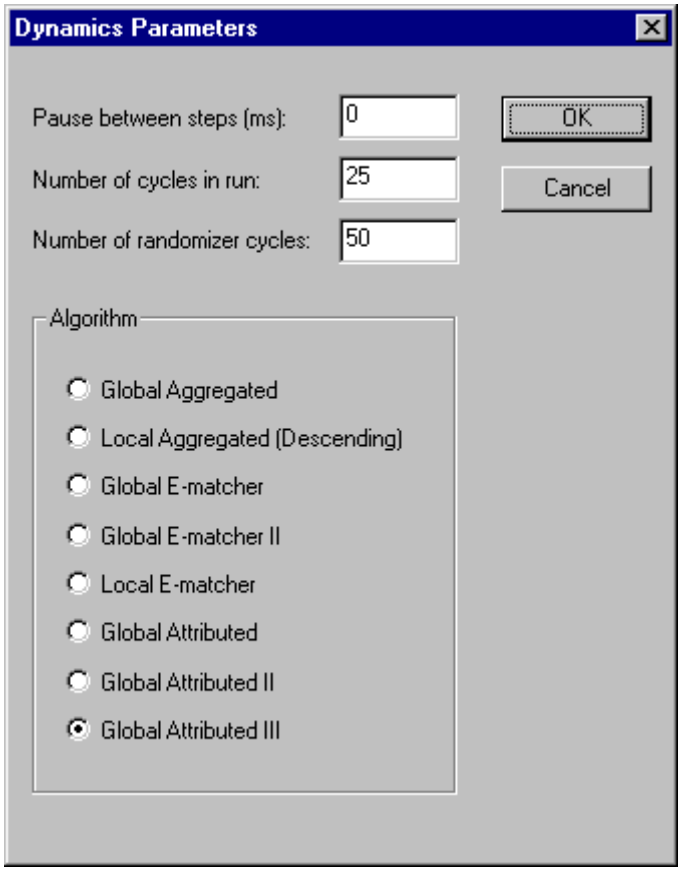


Figure 5 : La fenêtre « Dynamics Parameters ».

Huit algorithmes différents peuvent être choisis. Les algorithmes non documentés dans le mémoire de thèse sont d'une moindre importance. Notons que certains algorithmes ne sont définis que pour des SMAM augmentés. D'autres algorithmes nécessitent la représentation agrégée des agents. Répétons que cette représentation peut être activée et désactivée avec la commande « Dynamics / Aggregate ».

Entre chaque action effectuée par l'algorithme, le système se pause pendant la période spécifiée dans le champ « Pause between steps (ms) ». La période est exprimée en millisecondes. Le nombre de cycles à effectuer est spécifié dans le champ « Number of cycles in run ». Lorsqu'on souhaite effectuer une seule action de l'algorithme sélectionné, la commande « Dynamics / Run One Cycle » peut être utilisée.

La commande « Dynamics / Randomize » permet d'effectuer un nombre de migrations aléatoires dans le SMAM actif. Le nombre peut être spécifié en rédigeant le champ « Number of randomize cycles » dans la fenêtre « Dynamics Parameters ».

L'évolution de l'entropie d'un SMAM peut être visualisée en sélectionnant « View / History ». Cette sélection affiche une fenêtre secondaire comme celle de la figure 6. Le titre de la fenêtre indique quel SMAM (document) est représenté. La fenêtre peut être cachée en appuyant sur le bouton « OK » ou sur le bouton « X » en haut à droite de la fenêtre.

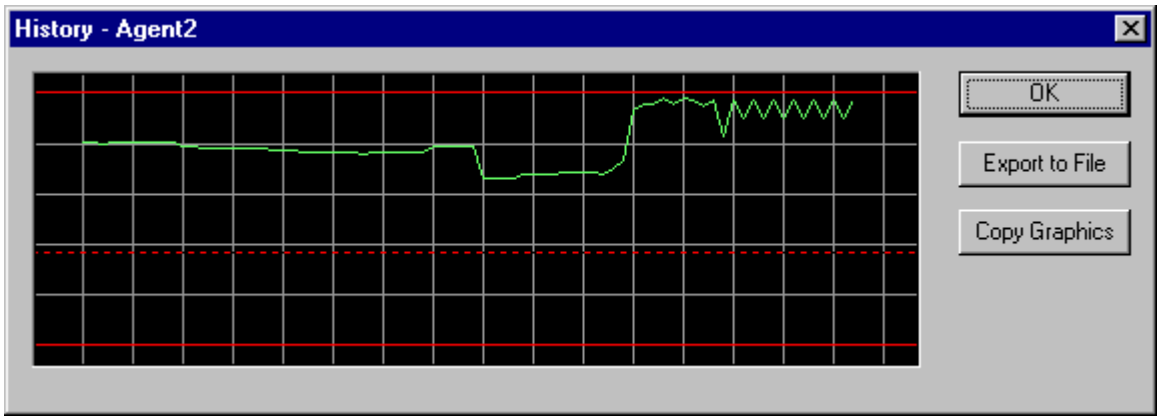


Figure 6 : La fenêtre « History ».

5. Afficher des SMAM

La visualisation des SMAM est une des fonctions principales de FRIENDS Offline. Les représentations supportées sont la représentation arborescente, la représentation triangulaire et la représentation « Arbre H ». Ces représentations sont conformes aux directrices décrites dans le mémoire de thèse. En particulier, l'agent le plus grand d'un couple est visualisé a gauche de son partenaire. Par contre, si on le souhaite, les couleurs utilisées peuvent être personnalisées.

La représentation arborescente

Dans la fenêtre principale, les SMAM sont visualisés à l'aide de la représentation arborescente. La commande « Dynamics / Aggregate » permet de choisir entre une représentation compacte et non compacte. En sélectionnant « View / Entropies », les valeurs des entropies des agent peuvent être affichées ou cachées. Cette commande est dupliquée par le bouton droit de la souris. La grille de la représentation arborescente peut être affichée ou caché en sélectionnant « View / Gridlines ». Comme nous l'avons déjà indiqué, l'affichage automatique des mots-clés dans un SMAM augmenté peut être activée ou désactivée en sélectionnant « View / Keywords ».

Si on le souhaite, on peut personnaliser les couleurs utilisés en sélectionnant « View / Colors... ». Notons toutefois que nous conseillons de respecter, dans des publications, les normes de visualisation spécifiées dans le mémoire de thèse.

La représentation triangulaire

Une fenêtre secondaire montrant la représentation triangulaire du SMAM actif peut être affichée en sélectionnant « View / Triangles ». La figure 7 montre un exemple d'une telle fenêtre. Le titre de la fenêtre indique quel SMAM (document) est représenté. Notons que, si la représentation agrégée est activée, la fenêtre secondaire visualise *tous* les SMAM atomiques d'ordre supérieur comme des SMAM atomiques d'ordre 0. La fenêtre peut être cachée en appuyant sur le bouton « OK » ou sur le bouton « X » en haut à droite de la fenêtre.

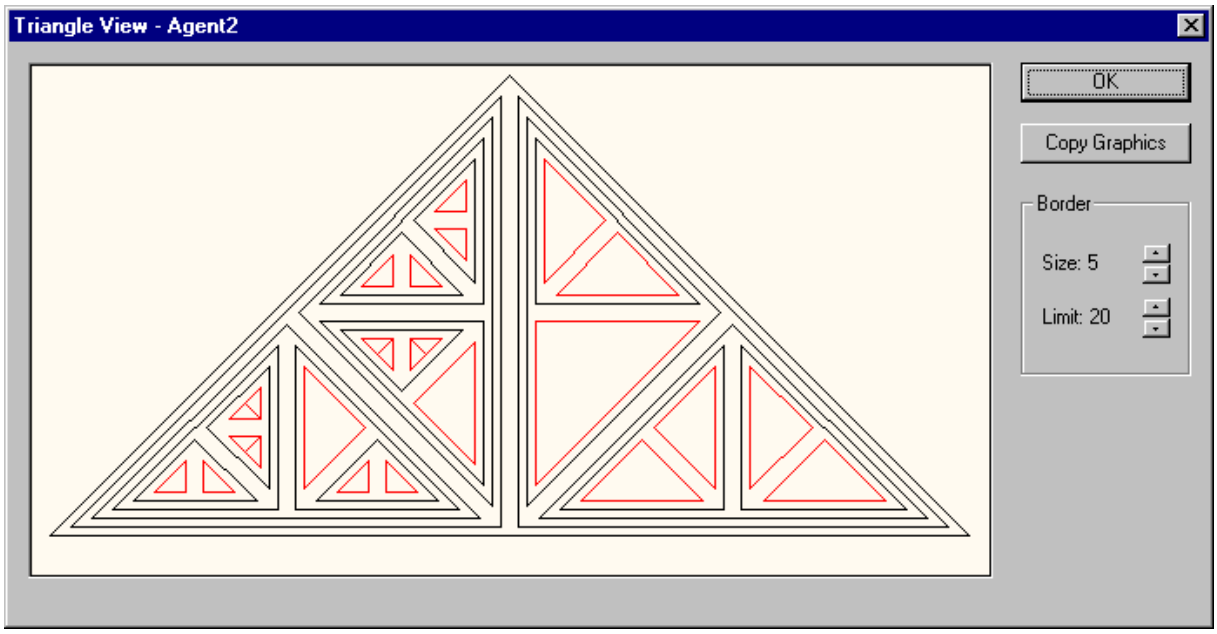


Figure 7 : La fenêtre « Triangle View ».

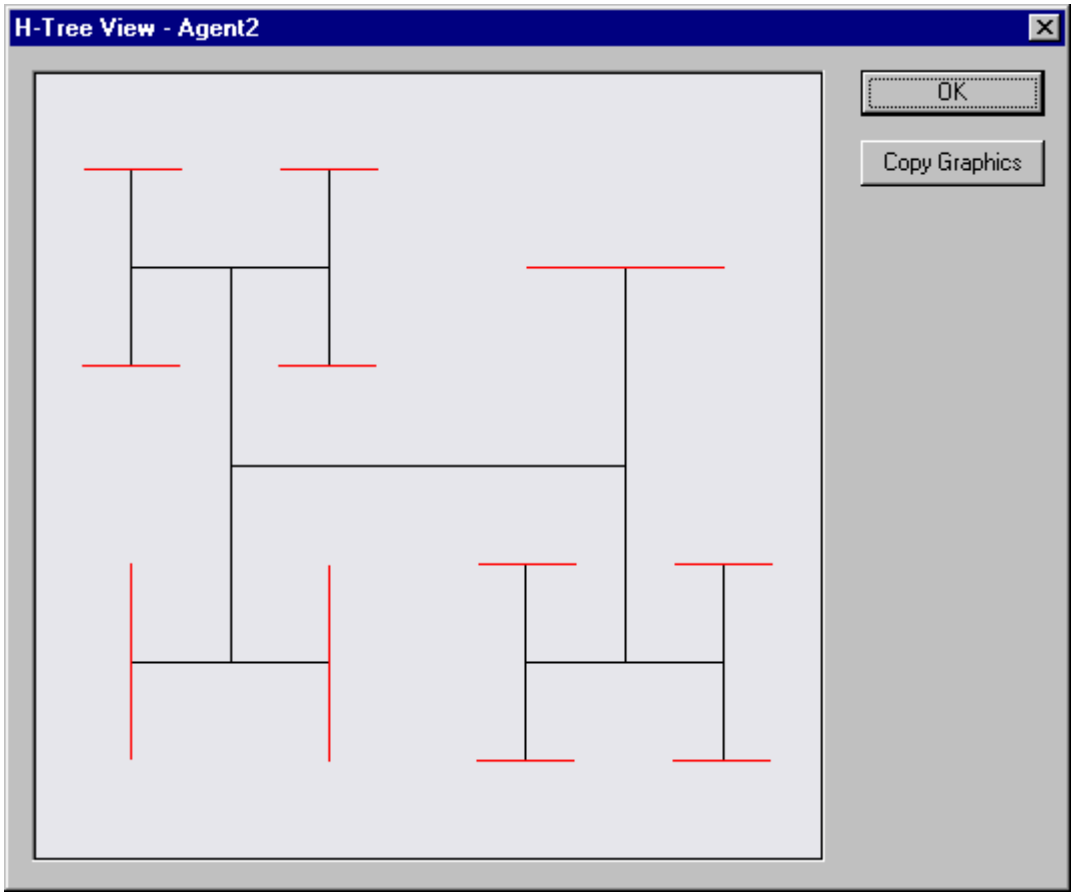


Figure 8 : La fenêtre « H-Tree View ».

Dans la fenêtre « Triangle View », le contrôle « Size » permet de modifier la taille de la bordure entre les agents de niveaux adjacents. La taille limite des triangles à partir de laquelle la bordure est mise à 0 peut être modifiée à l'aide du contrôle « Limit ».

La représentation « arbre H »

La sélection « View / H-Trees » permet d'afficher une fenêtre secondaire montrant la représentation « Arbre H » du SMAM actif. Une telle fenêtre est illustrée dans la figure 8. Le titre de la fenêtre indique quel SMAM (document) est représenté. Notons que, si la représentation agrégée est activée, la fenêtre secondaire visualise *tous* les SMAM atomiques d'ordre supérieur comme des SMAM atomiques d'ordre 0. La fenêtre peut être caché en appuyant sur le bouton « OK » ou sur le bouton « X » en haut à droite de la fenêtre.

6. Exporter et imprimer

Les représentations visuelles générées par FRIENDS Online peuvent facilement être copiées dans le presse-papiers de Windows, et être utilisées dans d'autres applications. Par exemple, une fois copiée dans le presse-papiers, la représentation d'un SMAM peut facilement être collée dans un document de Microsoft Word. Puisque la description graphique elle-même est copiée plutôt que le *bitmap* généré, une grande qualité de reproduction peut être atteinte.

La représentation arborescente de la fenêtre principale peut être copiée dans le presse-papiers en sélectionnant « Edit / Copy Graphics ». Les représentations triangulaires et « arbre H » peuvent être copiées en appuyant sur le bouton « Copy Graphics » dans les fenêtres secondaires.

De même, le graphique de la fenêtre « History » peut être copiée en appuyant sur le bouton « Copy Graphics » de cette fenêtre. Les données mêmes peuvent également être exportées. En appuyant sur le bouton « Export to File », un fichier peut être spécifié dans lequel les données sont écrites. Notons que les données décrivent l'évolution *complète*, à partir de la création du document ou de la dernière « Reset History ». L'extrait ci-dessous illustre le format texte utilisé :

```
Step      Min E       Max E       E
0         2.000000    6.546875    2.931055
1         2.000000    6.546875    3.863087
2         2.000000    6.546875    3.879025
3         2.000000    6.546875    3.912899
4         2.000000    6.546875    4.043149
5         2.000000    6.546875    4.044188
...
```

Les fichiers de ce format peuvent facilement être importés dans des applications tels que Microsoft Excel.

FRIENDS Offline permet d'imprimer directement la représentation arborescente de la fenêtre principale. Les commandes « Agent / Print... », « Agent / Print Preview » et « Agent / Print Setup » implémentent les fonctions d'impression classique de Windows.

7. Fonctions variées

En plus des fonctions décrites dans les sections précédentes, quelques autres sont supportées par FRIENDS Offline.

La commande « Agent / Exit » permet de quitter l'application. « View / Toolbar » et « View / Statusbar » permettent d'afficher ou de cacher respectivement la barre d'outils et la barre d'état. Le menu « Window » offre quelques fonctions classiques des applications MDI de Windows. En sélectionnant « Window / Cascade », les fenêtres sont affichées les unes sur les autres. En choisissant « Window / Tile », les fenêtres sont affichées côte à côte. Une deuxième série de commandes de ce menu peut être utilisée pour sélectionner rapidement une fenêtre.

La sélection de « Numbers / Possible Organisations » affiche la fenêtre de la figure 9. Ces données peuvent également être écrites dans un fichier.

Enfin, « Help / About FRIENDS... » affiche une fenêtre contenant quelques informations sur l'application.

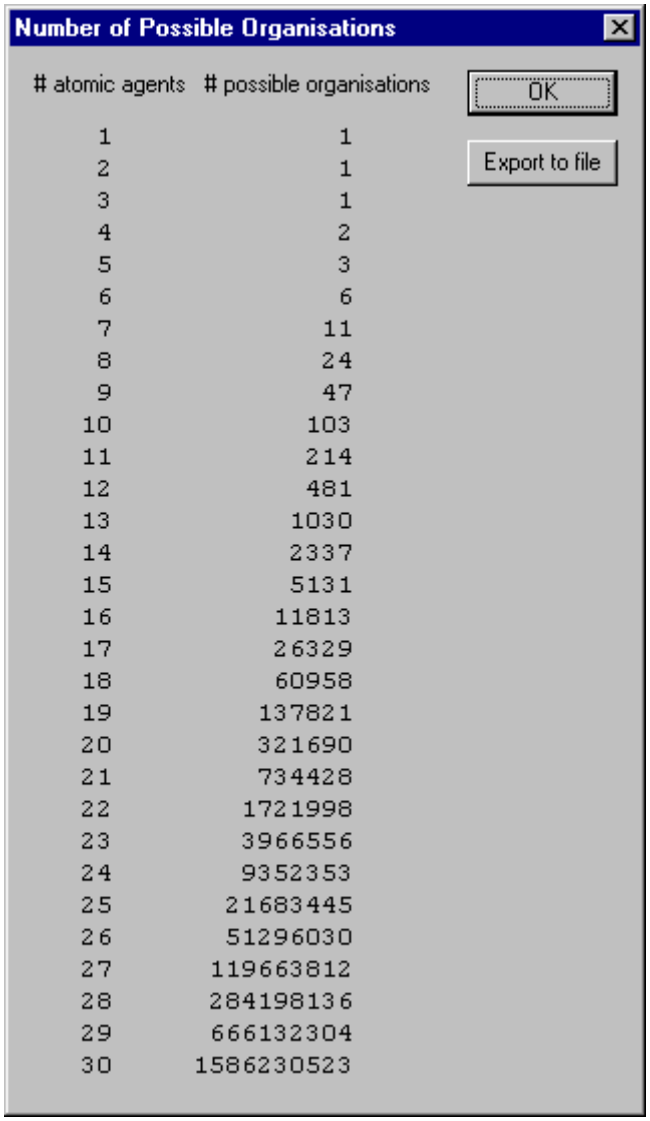


Figure 9 : La fenêtre « Number of Possible Organisations ».

8. Bogues — fonctions non implémentées

Dans notre expérience, FRIENDS Offline ne contient pas de bogues. Toutefois, nous avons identifié un inconvénient que nous souhaitons corriger. Lorsqu'on essaie de faire évoluer un SMAM pur par un algorithme défini uniquement pour des SMAM augmentés, un message d'erreur est affiché pour *chaque* action effectuée. Nous comptons élimer cet inconvénient dans la version 1.8 du programme.

Les commandes de sauvegarde « Agent / Open », « Agent / Save » et « Agent / Save as... » ne sont pas encore implémentées. Nous envisageons de les incorporer dans la version 1.8. *Notons que la commande « Agent / Open » ne doit pas être essayée sur un fichier arbitraire. Ceci peut provoquer la terminaison du programme.*

De même, la commande « Help / Help Topics » n'est pas implémentée. Nous souhaitons supporter l'aide en ligne dans la version 1.8.

La commande « Dynamics / Stop » est désactivée. Dans un premier temps, nous n'envisageons pas de supporter cette fonction.

ANNEXE IV

FRIENDS NUMBERCRUNCHER

1. Introduction

Cette annexe constitue un manuel compact de FRIENDS Numbercruncher 1.0. Le logiciel nécessite Windows NT 4.0, Windows 95 ou Windows 98. La seule langue actuellement supportée est l'anglais. La taille de la version autonome (*Statically Linked*) du code exécutable est 201 Ko.

L'application ne nécessite pas de procédure d'installation et peut directement être lancée en double-cliquant sur son icône, montré dans la figure 1.



Figure 1 : L'icône de FRIENDS Numbercruncher 1.0.

L'utilisation du programme se décompose en 5 phases, nommées « Start Analysis », « Build Dictionary », « Create Population », « Start Organisation » et « Generate Report ». Chaque phase est initiée en appuyant sur le bouton correspondant. Parcourons ces différentes étapes.

2. Première phase : « Start Analysis »

Lorsque l'on lance FRIENDS Numbercruncher, la fenêtre de la figure 2 est affichée. La première phase est initiée en appuyant sur le bouton « Start Analysis ». Dans cette phase, le programme invite l'utilisateur à charger un fichier de texte contenant la description de la population. Le format de ce fichier doit être conforme au format spécifié par le CNET. Dans ce format, les mots-clés sont séparés par des espaces ou des fins de lignes. Les listes différentes, correspondant aux différents agents, sont séparées par des lignes vides. Illustrons ces conventions en présentant le contenu d'un fichier (d'une taille très modeste) conforme au format :

```
chat chien plage vacances voyages soleil
poissons vacances animaux

ordinateurs Internet Web
Bill-Gates email photographie PC Macintosh Intel
cryptographie complexité chaos

systèmes-dynamiques chaos Vie-Artificielle entropie
photographie poissons Seychelles voyages
```

Ce fichier contient les listes des mots-clés de trois agents. Si un « mot-clé » est composé de plusieurs mots, les composants sont liés par un tiret. Notons que les listes peuvent contenir des mots-clés en double.

Une fois le fichier chargé, son nom, le nombre d'agents et le nombre de mots-clés (incluant les doubles) sont affichés. Les mots-clés (les double inclus) sont affichés aussi, par ordre alphabétique.

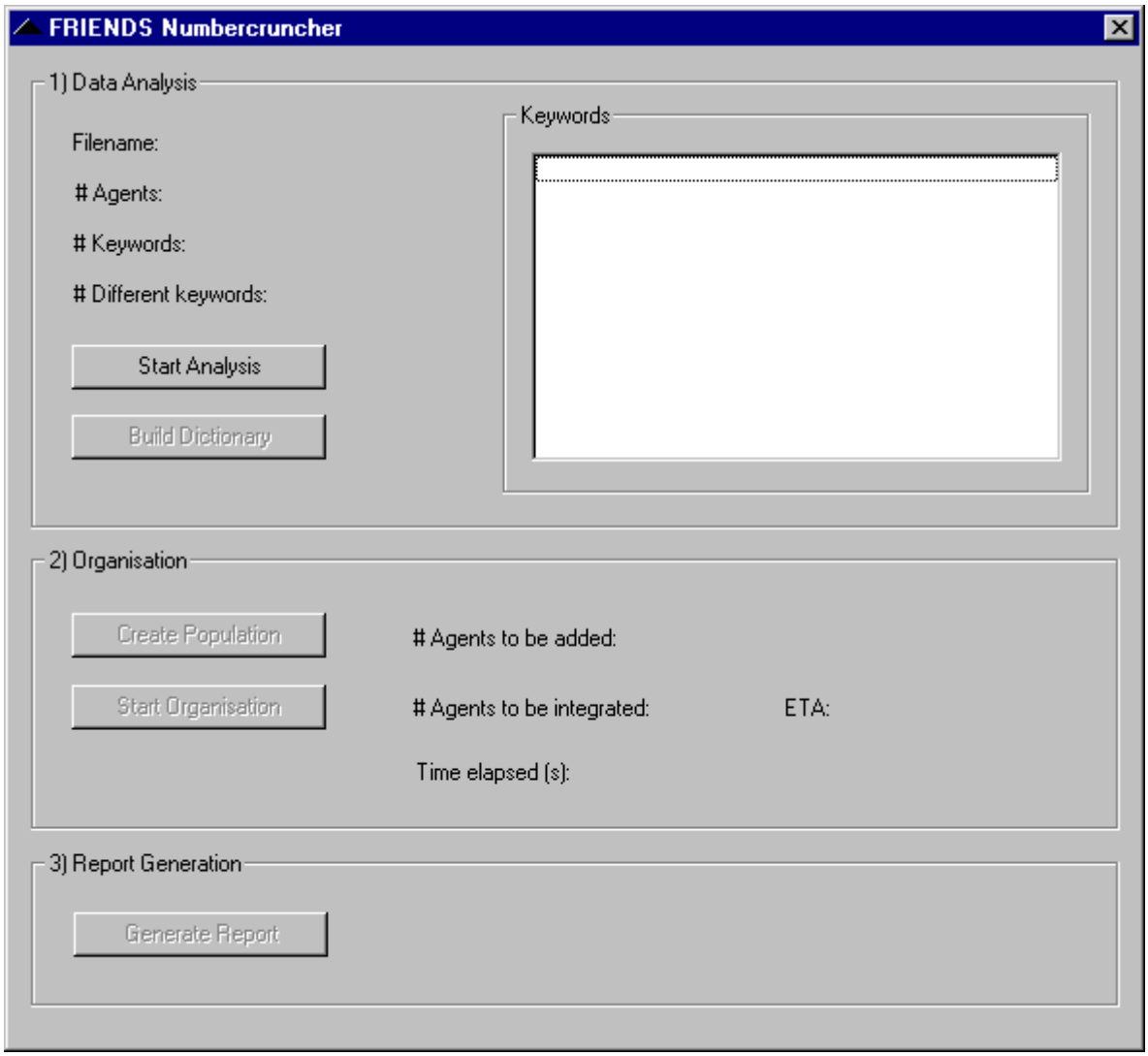


Figure 2 : Début de la première phase.

3. Deuxième phase : « Build Dictionary »

La figure 3 montre un exemple de la fenêtre de FRIENDS Numbercruncher au début de la deuxième phase. Dans cette étape, initiée en appuyant sur le bouton « Build Dictionary », un dictionnaire global est construit, liant chaque mot-clé à un nombre entier. Si deux mots-clés sont lexicalement identiques, ils sont liés au même nombre, éliminant les doubles. L'utilisation d'un dictionnaire accélère sensiblement l'organisation de la population. Notons que, si le nombre de mots-clés est grand, la construction du dictionnaire peut prendre un certain temps.

Après la construction, le nombre de mots-clés différents est affiché. Les mots-clés (sans doubles et par ordre alphabétique) sont également affichés.

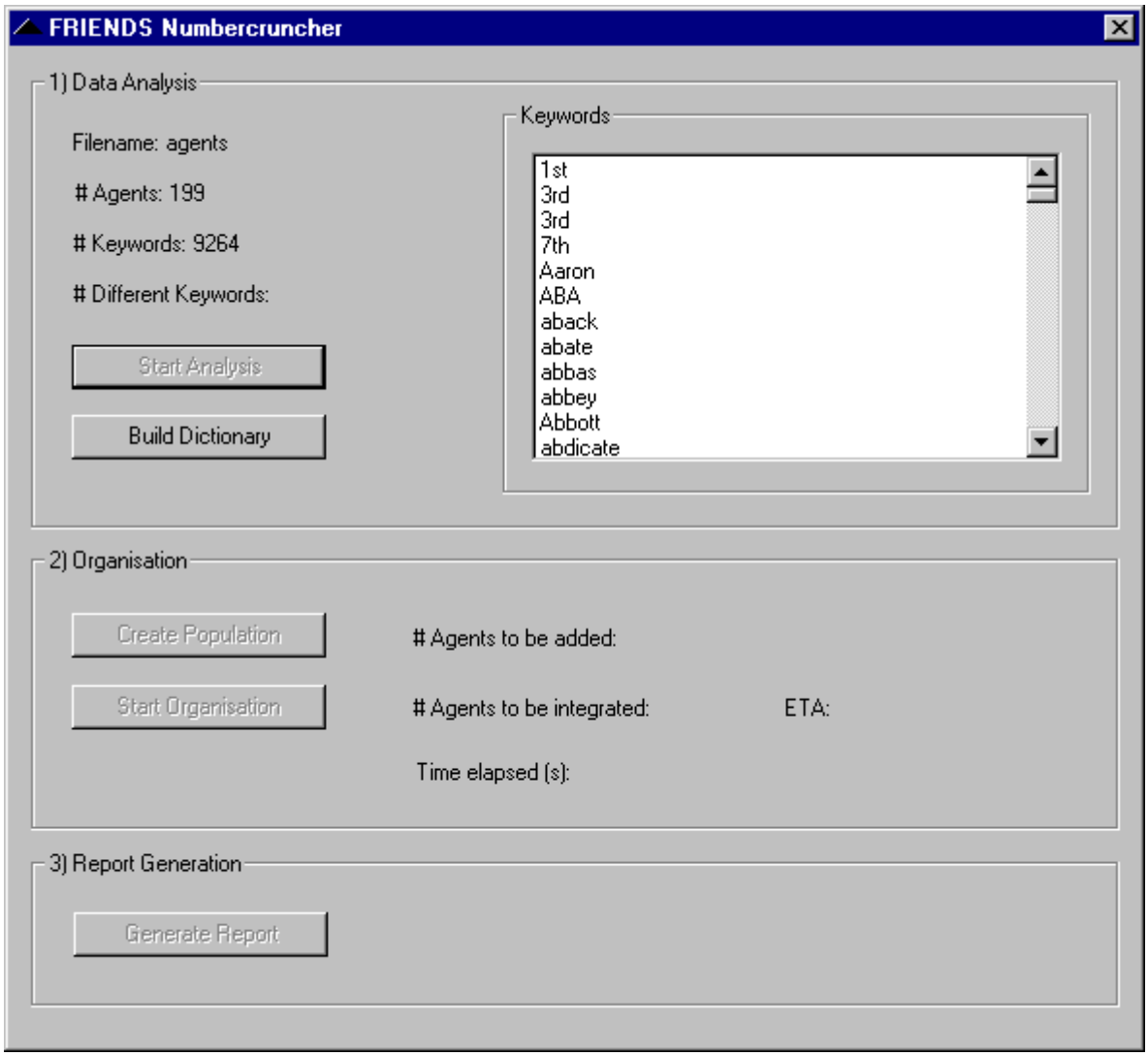


Figure 3 : Début de la deuxième phase.

4. Troisième phase : « Create Population »

Une fois la deuxième phase finie, la fenêtre de l'application ressemble à celle de la figure 4. En appuyant sur le bouton « Create Population », la troisième phase est initiée.

Dans cette phase, une représentation optimale - en considérant le temps de calcul - de la population est créée. La liste de mots-clés de chaque agent est convertie, en utilisant le dictionnaire global, en une liste de nombres entiers tous différents. Si le nombre total de mots-clés est grand, ce processus aussi peut prendre un certain temps. Le nombre d'agent non encore ajoutés à la population est affiché, permettant le suivi du progrès.

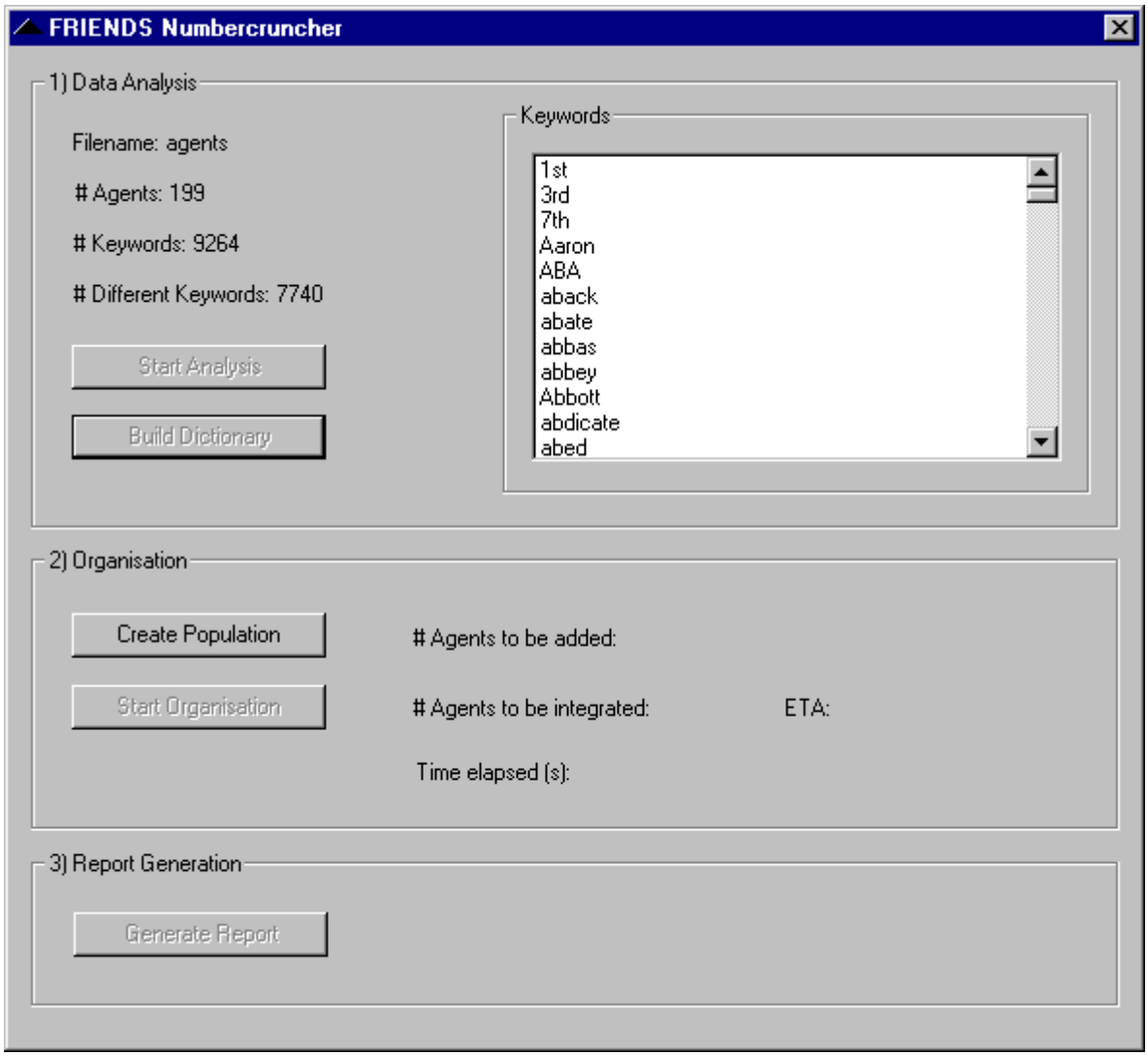


Figure 4 : Début de la troisième phase.

5. Quatrième phase : « Start Organisation »

Une fois la population créée, la fenêtre de l'application ressemble à celle de la figure 5. L'organisation de la population, constituant la quatrième phase, est initiée en appuyant sur le bouton « Start Organisation ». Durant l'organisation s'affichent le nombre d'agents non encore intégrés, le temps écoulé et le « Estimated Time of Arrival ».

L'algorithme utilisé est de complexité $O(n^3)$ et peut nécessiter, pour des grandes populations, un temps considérable. Comme indication, le tableau 1 montre des temps d'organisation pour trois populations de tailles différentes. Les résultats sont obtenus sur un Dell Dimension XPS D300 (à base d'un Pentium II à 300 MHz) sous Windows NT 4.0. Dans ces essais, chaque agent a - en moyenne - 10 mots-clés.

Lorsqu'on double le nombre d'agents, le temps est augmenté par un facteur 10, plutôt que par le facteur 8 attendu. La baisse d'efficacité des *caches* du processeur (en fonction de la taille des données) peut être à l'origine de ce phénomène. Pour la même raison, le *Estimated Time of Arrival*, affiché par le programme est souvent un peu pessimiste (car l'estimation est basée sur le temps déjà écoulé).

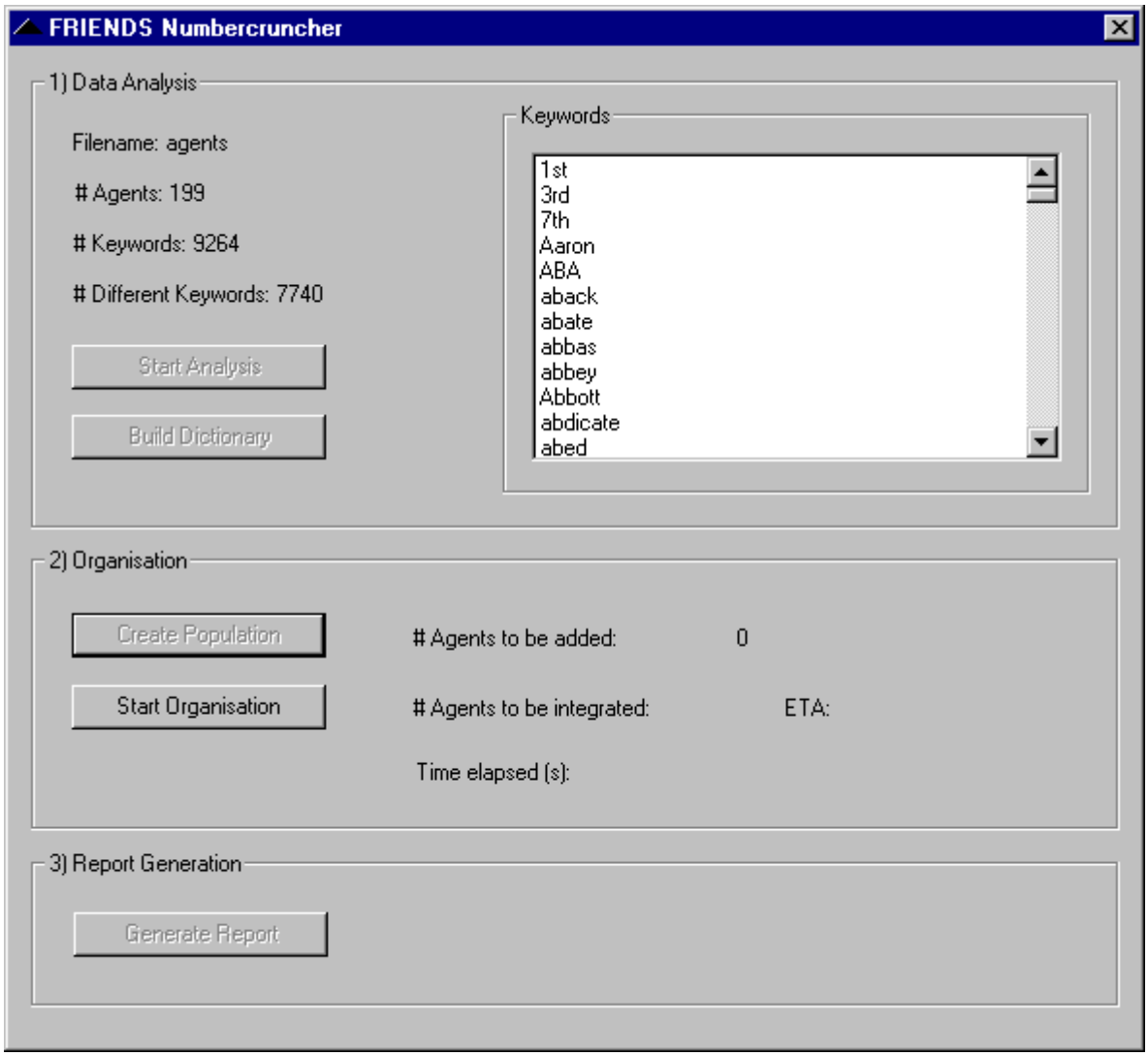


Figure 5 : Début de la quatrième phase.

Taille de la population	**Temps d'organisation**
500 agents	20 secondes
1000 agents	219 secondes
2000 agents	2075 secondes

Tableau 1 : Temps d'organisation.

6. Cinquième phase : « Generate Report »

La figure 6 montre un exemple de la fenêtre de FRIENDS Numbercruncher après l'organisation des agents. L'étape finale, initiée en appuyant sur le bouton « Generate Report », consiste en la génération d'un rapport. Au début de cette phase, le programme invite l'utilisateur à spécifier le fichier dans lequel le rapport sera écrit.

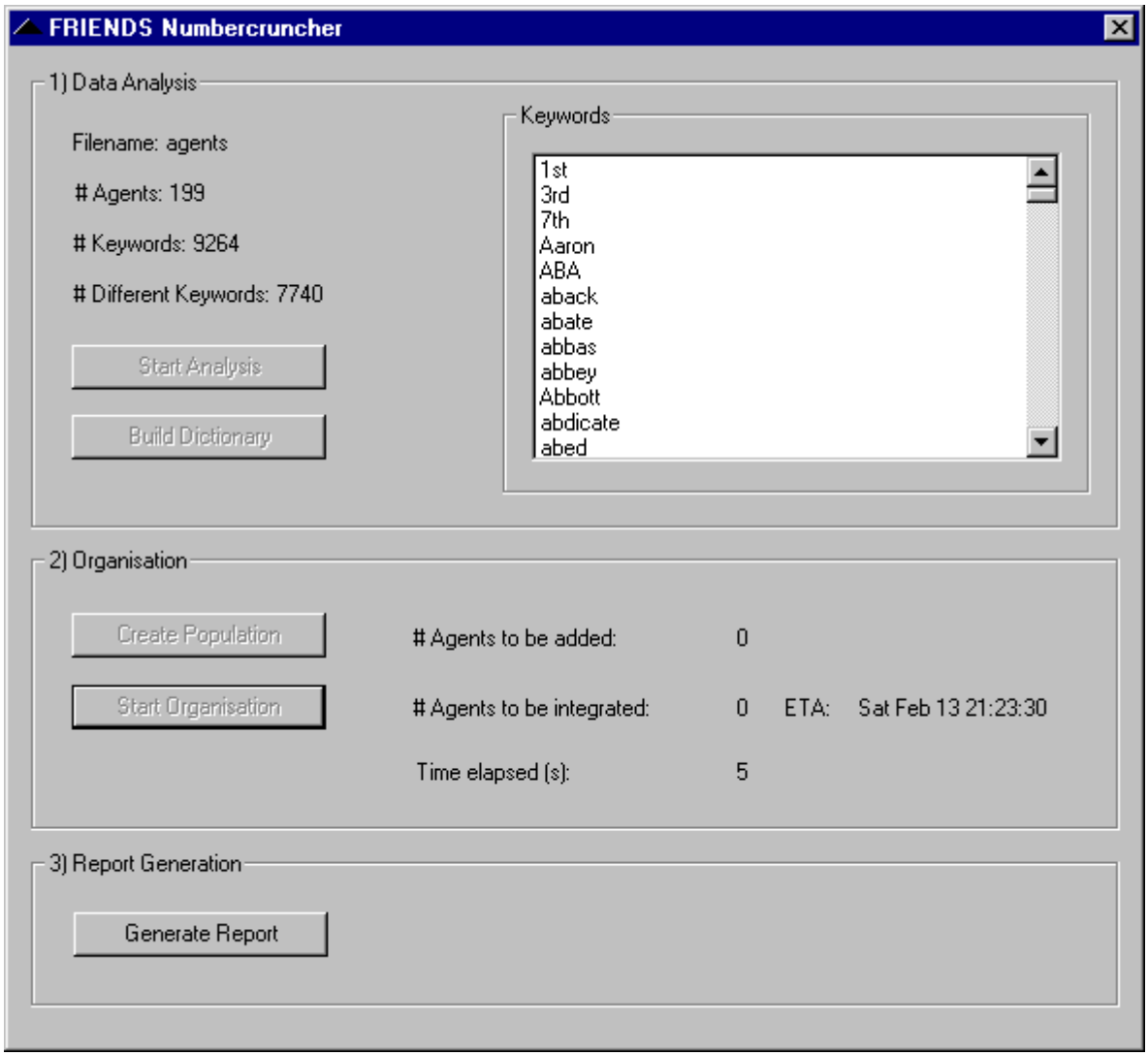


Figure 6 : Début de la cinquième phase.

Dans le rapport généré, le dendrogramme correspondant à la structure du SMAM final est représenté d'une manière alphanumérique. Chaque ligne représente un agent. A chaque ligne, la liste de mots-clés est précédée par un nombre de tirets, correspondant au niveau de l'agent. La présence d'une astérix avant la liste des mots-clés indique que l'agent est atomique.

Illustrons cette présentation à l'aide d'un exemple :

```
<ligne vide>
-* Bill-Gates chaos complexité cryptographie email Intel Internet
   Macintosh ordinateurs PC photographie Web
-  poissons voyages
--* animaux chat chien plage poissons soleil vacances voyages
--* chaos entropie photographie poissons Seychelles systèmes-
    dynamiques Vie-Artificielle voyages
```

Notons que, dans cet exemple, certaines lignes dépassent la largeur de la page et ont du être tronquées.

7. Une nouvelle analyse

La figure 7 présente un exemple de la fenêtre de l'application après la génération du rapport. Notons que le bouton « Start Analysis » est de nouveau actif. En effet, Une fois un rapport généré, une nouvelle analyse peut être effectuée.

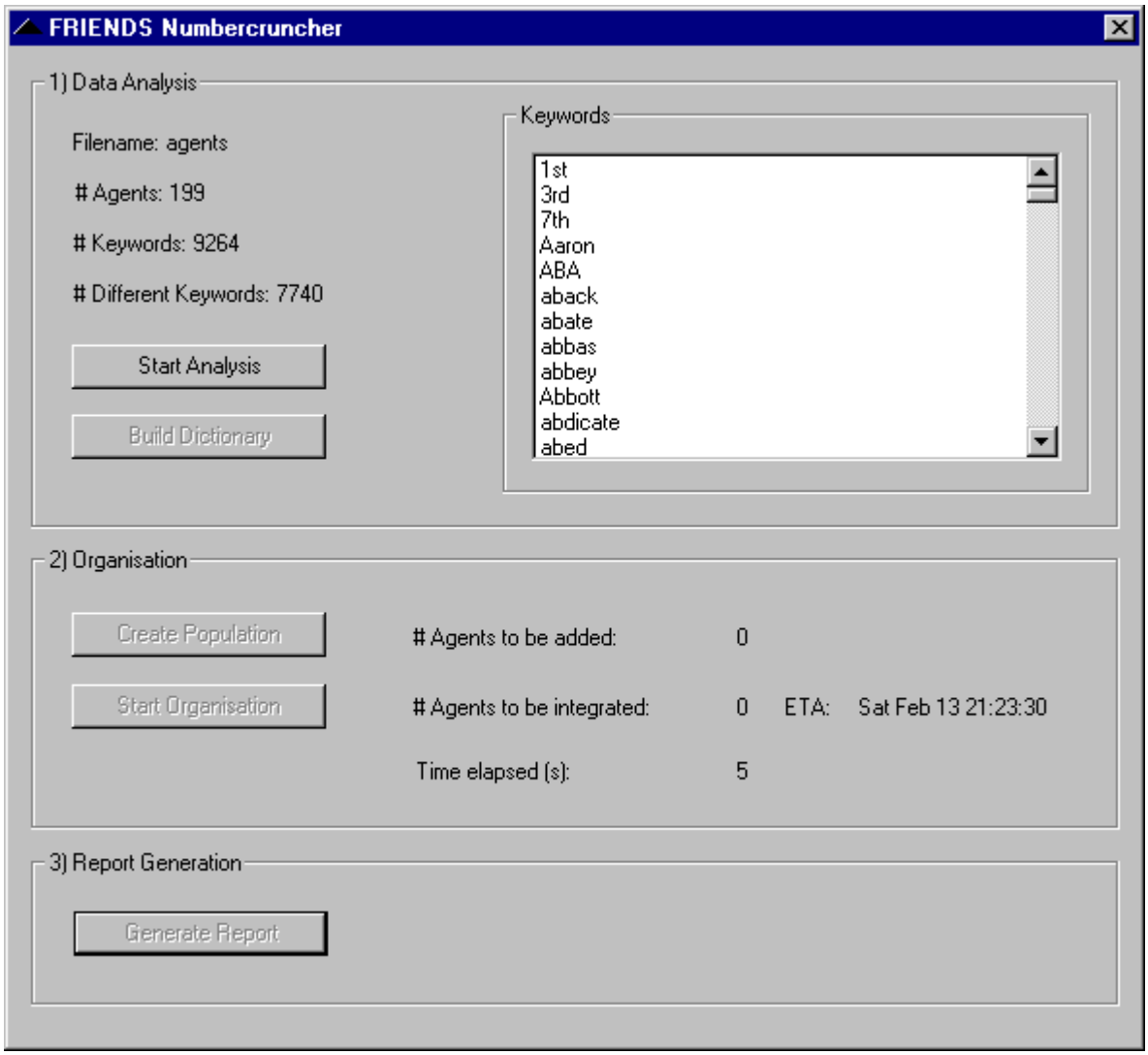


Figure 7 : Fin de la cinquième phase.

8. Bogues

Dans notre expérience, FRIENDS Numbercruncher ne contient pas de bogues. Toutefois, notons que le système est conçu pour Windows NT 4.0, plutôt que pour Windows 95 et Windows 98. L'affichage dans la fenêtre de l'application d'un très grand nombre de mots-clés peut être problématique sous Windows 95 et Windows 98.

ANNEXE V

FRIENDS ONLINE

1. Introduction

Cette annexe constitue un manuel compact de FRIENDS Online 3.1. Le logiciel nécessite le Java Development Kit 1.1. La seule langue actuellement supportée est l'anglais. La somme des tailles des fichiers `.class` est 90 Ko.

L'application est constituée de trois parties : l'application « Server », l'applet « Administrator » et l'applet « Client ». Les deux applets peuvent être exécutées comme des applications.

2. l'application « Server »

Le serveur est lancé en effectuant la commande « java FriendsServer ».

Il n'y a pas d'interaction directe avec le serveur. Toute interaction passe par les deux applets.

3. L'applet « Administrator »

Cette applet peut être lancée en chargeant la page HTML correspondante ou en effectuant la commande « java FriendsAdministrator ».

La figure 1 montre sa fenêtre initiale.

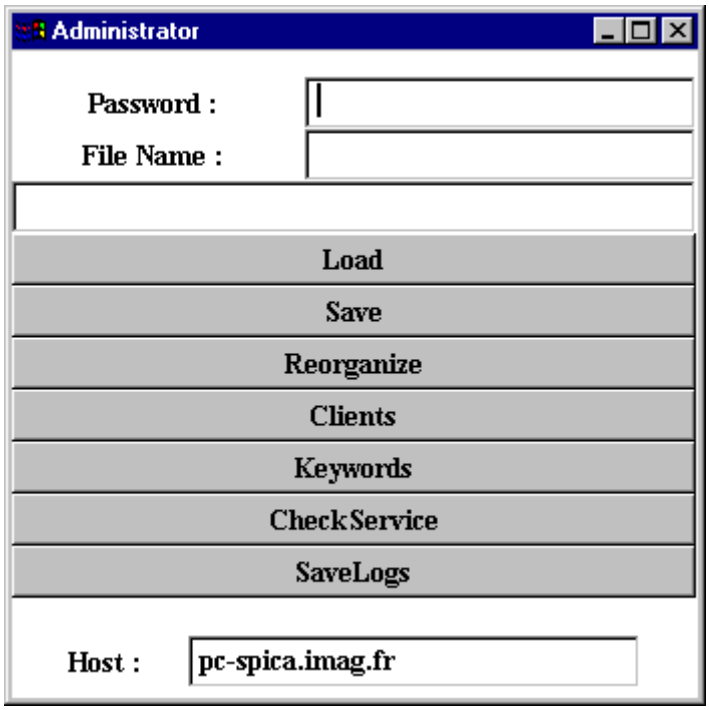


Figure 1 : La fenêtre « Administrator » initiale.

L'utilisation de cette applet nécessite la spécification d'un mot de passe. Dans la version de FRIENDS Online fournie avec ce mémoire, le mot de passe est « abcdef ».

Une fois le mot de passe spécifié, les fonctions de l'applet sont accessibles. Toutefois, avant toute action, l'hôte sur lequel se trouve le serveur doit être spécifié dans le champ « Host ». Notons qu'un hôte ne peut héberger qu'un seul serveur.

Le champ au-dessus du bouton « Load » affiche des informations concernant les actions effectuées.

Parcourons les différentes fonctions.

La fonction « Load »

En spécifiant un nom de fichier dans le champ « File Name » et en appuyant sur le bouton « Load », le fichier correspondant, décrivant un SMAM de type FRIENDS, est chargé par le serveur spécifié.

La figure 2 montre un exemple de la fenêtre de l'applet après le chargement d'un fichier.

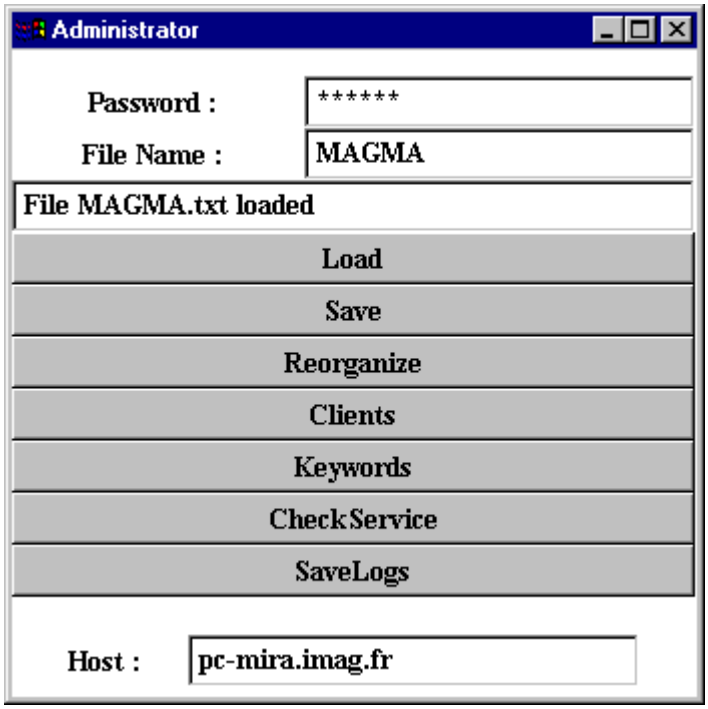


Figure 2 : La fenêtre « Administrator » après le chargement d'un fichier.

La fonction « Save »

En spécifiant un nom de fichier dans le champ « File Name » et en appuyant sur le bouton « Save », le SMAM du serveur spécifié est sauvegardé.

La fonction « Reorganize »

Lorsque l'on appuie sur le bouton « Reorganize », le SMAM du serveur est globalement réorganisé. Pendant la réorganisation, aucune modification du SMAM est possible.

La fonction « Clients »

En appuyant sur le bouton « Clients », la fenêtre « Clients of Administrator » est affichée, permettant la gestion des clients du serveur. La figure 3 montre un exemple d'une telle fenêtre.

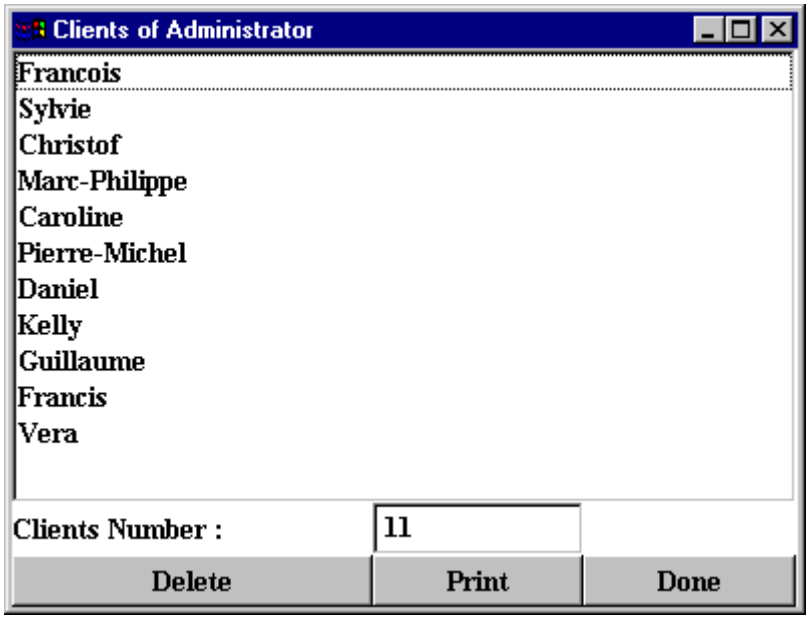


Figure 3 : La fenêtre « Clients of Administrator ».

La fonction « Keywords »

Lorsque l'on appuie sur le bouton « Keywords », une fenêtre s'affiche permettant la gestion des mots-clés du SMAM du serveur. La figure 4 montre un exemple d'une telle fenêtre.

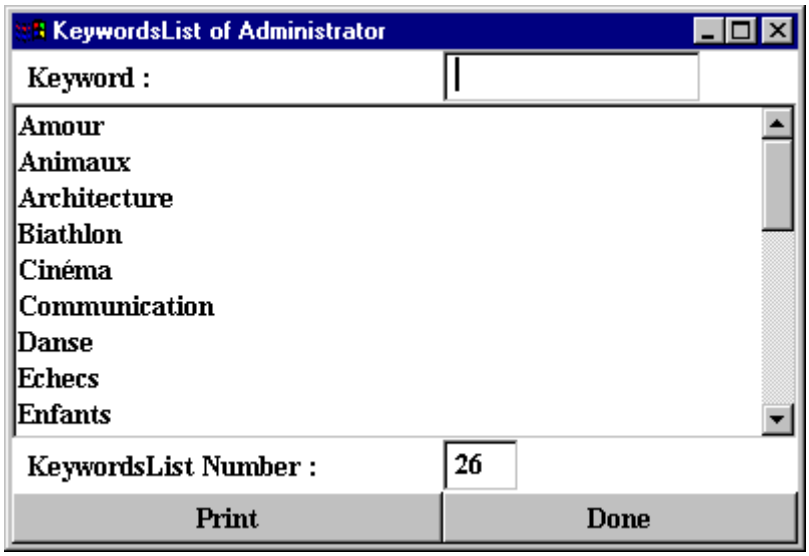


Figure 4 : La fenêtre « KeywordsList of Administrator ».

La fonction « CheckService »

L'état du serveur peut être vérifié en appuyant sur le bouton « CheckService ». Cette action affiche une fenêtre indiquant si le serveur est en état de réorganisation globale ou non. La figure 5 présente la fenêtre affichée lorsque le serveur est en train de réorganiser le SMAM.

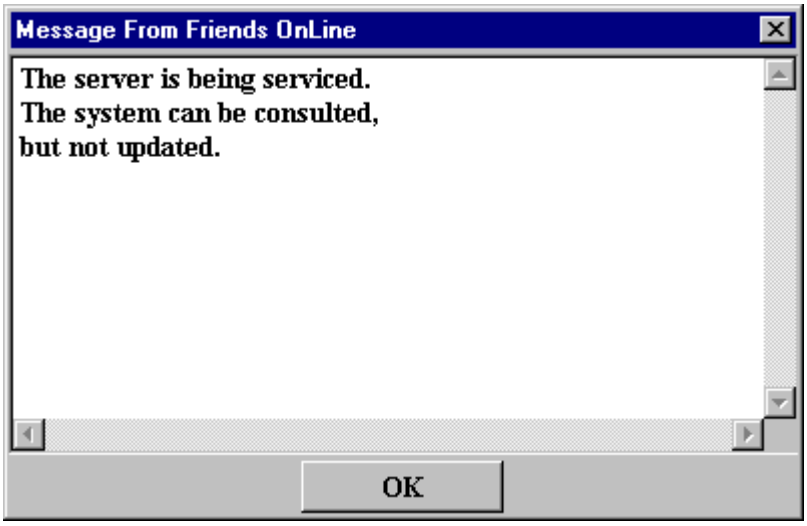


Figure 5 : La fenêtre « Message From Friends OnLine ».

La fonction « SaveLogs »

En spécifiant un nom de fichier dans le champ « File Name » et en appuyant sur le bouton « SaveLogs », l'historique du système peut être sauvegardée.

4. L'applet « Client »

Cette applet peut être lancée en chargeant la page HTML correspondante ou en effectuant la commande « java FriendsClient ». Si l'applet est lancée comme application, une fenêtre apparaît demandant le nom de l'hôte hébergeant le serveur. La figure 6 présente un exemple de cette fenêtre.



Figure 6 : La fenêtre « Host for use Friends OnLine ».

Une fois l'hôte spécifié (ou la page chargée), une fenêtre s'affiche permettant de rentrer nom et mot de passe. Si on ne spécifie pas ces deux éléments, on peut observer le système sans y participer.

La figure 7 montre un exemple de cette fenêtre d'identification.

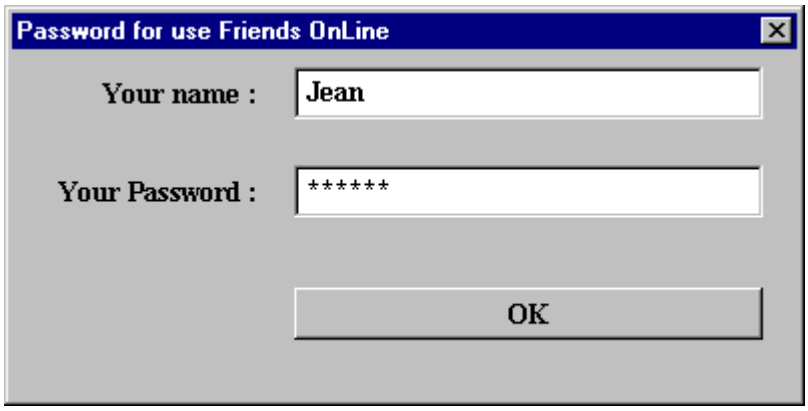


Figure 7 : La fenêtre « Password for use Friends OnLine ».

Si un nom inconnu du serveur est spécifié, l'applet demande à l'utilisateur de spécifier une liste de mots-clés décrivant ses intérêts. La figure 8 montre un exemple de la fenêtre utilisée.

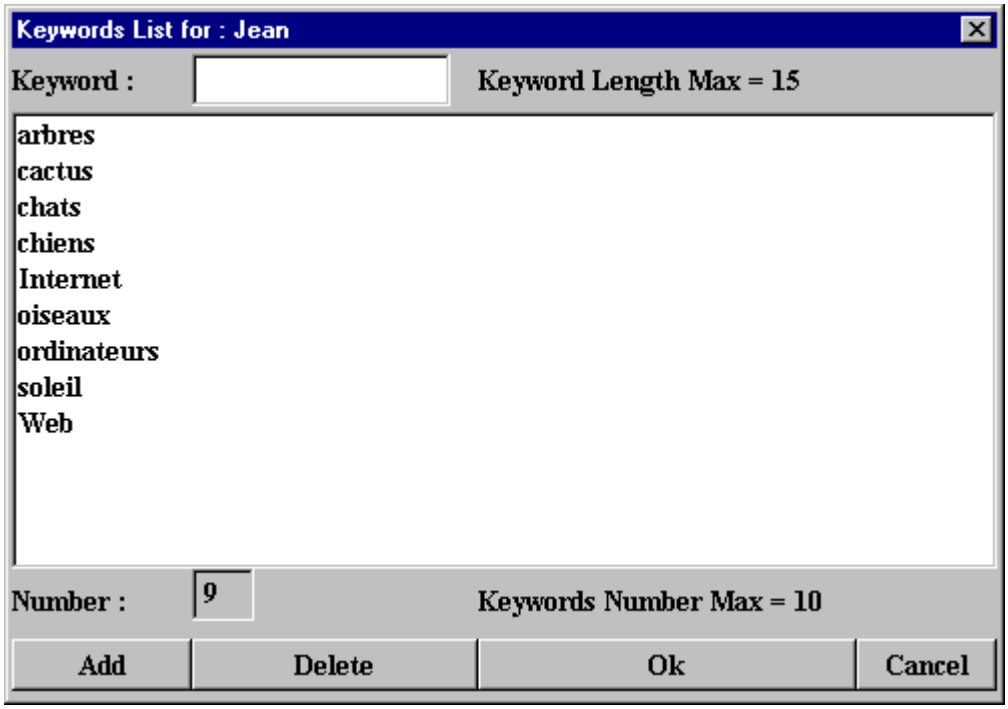


Figure 8 : La fenêtre « Keywords List for : Jean ».

Une fois la phase d'introduction terminée, la fenêtre principale de FRIENDS Online est affichée. La figure 9 montre un exemple d'une telle fenêtre.

En haut de la fenêtre sont présentés le nombre d'agents atomiques (utilisateurs), le niveau maximal des agents affichés, et les mots-clés hérités par tous les agents affichés.

Les agents atomiques sont visualisés en rouge, les agents composés en noir. L'agent représentant l'utilisateur même (s'il s'est identifié) est affiché en bleu.

En bas de la fenêtre se trouvent des boutons pour effectuer différentes manipulations. Parcourons-les.

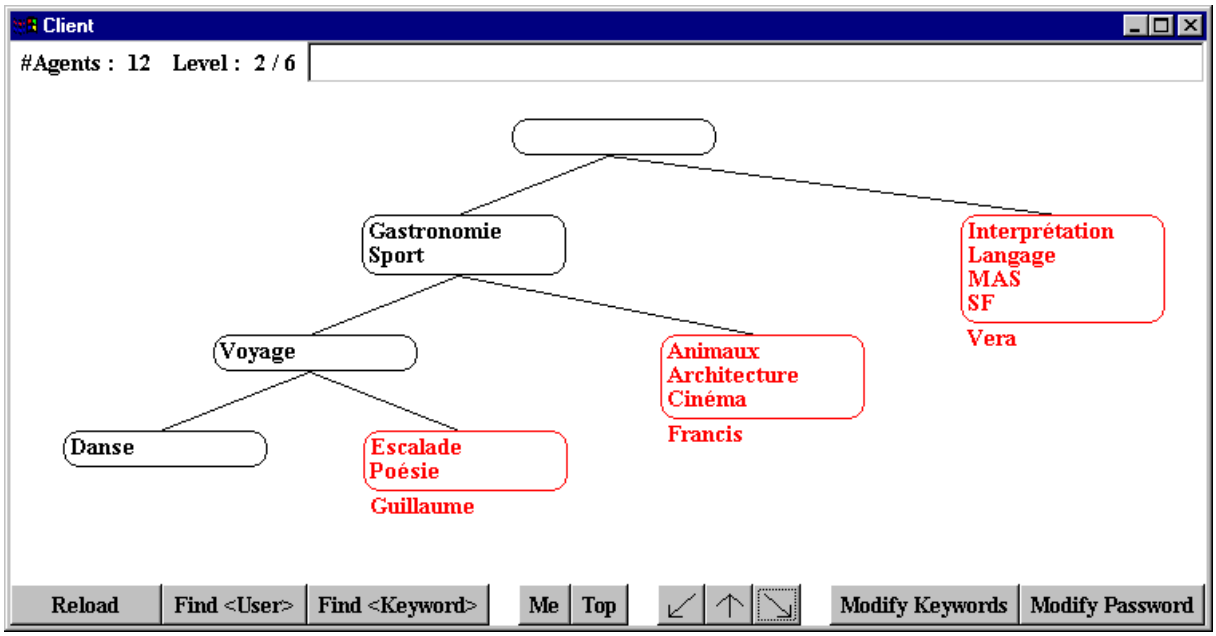


Figure 9 : La fenêtre « Client ».

La fonction « Reload »

En appuyant sur le bouton « Reload », l'applet met à jour le SMAM affiché en contactant le serveur.

La fonction « Find <User> »

Lorsqu'on appuie sur le bouton « Find <User> », une fenêtre de recherche est affichée, permettant de trouver un utilisateur dans le SMAM. La figure 10 montre un exemple d'une telle fenêtre. Si l'utilisateur existe, son agent est affichée.

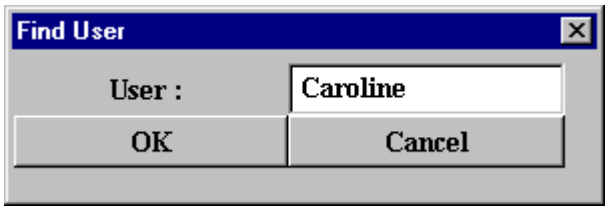


Figure 10 : La fenêtre « Find User ».

La fonction « Find <Keyword> »

En appuyant sur le bouton « Find <Keyword> », une fenêtre comme celle de la figure 11 est affichée, permettant de spécifier un mot-clé. Les résultats de la recherche sont présentés dans une fenêtre dédiée. La figure 12 montre un exemple d'une telle fenêtre.



Figure 11 : La fenêtre « Find Keyword ».

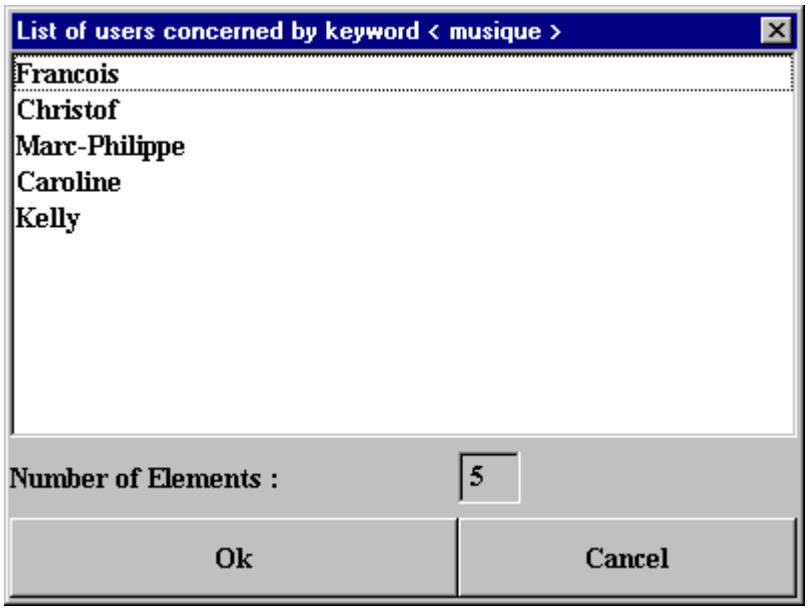


Figure 12 : La fenêtre « List of users concerned by keyword < musique > ».

La fonction « Me »

Lorsqu'on appuie sur le bouton « Me », on est positionné sur son propre agent atomique.

La fonction « Top »

En appuyant sur le bouton « Top », on est positionné au sommet de l'arborescence.

Les fonctions « Flèches »

Les trois boutons « flèches » permettent de naviguer dans l'arborescence.

La fonction « Modify Keywords »

Lorsqu'on appuie sur le bouton « Modify Keywords », une fenêtre comme celle de la figure 8 est affichée. A l'aide de cette fenêtre, on peut modifier ses mots-clés.

Notons que cette fonction n'est activée que lorsqu'on s'est identifié au système.

La fonction « Modify Password »

Un utilisateur régistré peut changer son mot de passe en appuyant sur le bouton « Change Password ». Cette action affiche une fenêtre comme celle de la figure 13, permettant le choix d'un nouveau mot de passe.

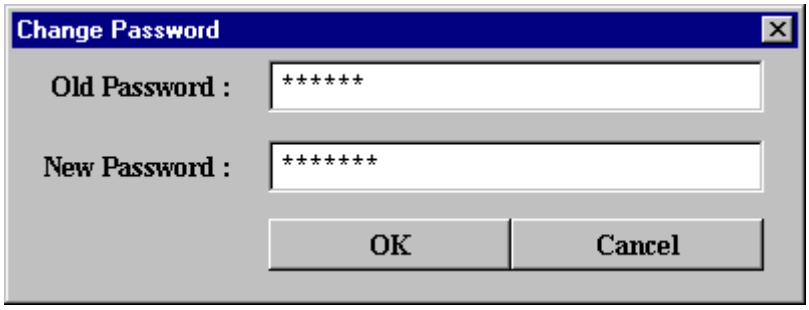


Figure 13 : La fenêtre « Change Password ».

ANNEXE VI

LES SYSTEMES MULTI-AGENTS MINIMAUX, LE CD

Un CD-ROM accompagne ce mémoire de thèse.

Il contient :

1. le mémoire de thèse en PostScript
2. le mémoire de thèse en PDF
3. le `.exe` de FRIENDS Offline
4. le `.exe` de FRIENDS Numbercruncher
5. les `.class` de FRIENDS Online

Plus d'informations se trouvent dans le fichier « LisezMoi.txt ».

Le CD est copyright 1999, INPG et France Télécom.

BIBLIOGRAPHIE

[Ackerman 96] Mark S. Ackerman, Brian Starr, « Social Activity Indicators for Groupware », *Computer*, volume 29, numéro 6, juin 1996, pp. 37-42.

[Adler 95] Richard M. Adler, « Emerging Standards for Component Software », *Computer*, volume 28, numéro 3, mars 1995.

[Aknine 98] Samir Aknine, Suzanne Pinson, Manuel Zacklad, « Un système d'agents récursifs pour l'aide au travail coopératif », *Actes des 6ièmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Pont-à-Mousson, France, novembre 1998.

[Arnold 98] K. Arnold, James Gosling, *The Java Programming Language*, Second Edition. Addison-Wesley, 1998.

[Baeijs 96] Christof Baeijs, Yves Demazeau, « Les Organisations dans les Systèmes Multi-Agents », *Actes de la 4ème Journée du GDR-PRC IA*, Toulouse, France, février 1996, pp. 35-46.

[Baeijs 97] Christof Baeijs, Yves Demazeau, « From Spatial Databases to Organized Multi-Agent Systems », *Proceedings of the 1st International Workshop on DAI and MAS*, Saint-Petersburg, 1997, pp. 54-63.

[Berghel 98] Hal Berghel, « Who Won the Mosaic War? », *Communications of the ACM*, volume 41, numéro 10, octobre 1998, pp. 13-16.

[Berlioux 88] Pierre Berlioux, Philippe Bizard, *Algorithmique 2, structures de données et algorithmes de recherche*, Bordas, Paris, France, 1988.

[Berlekamp 82] Elwyn R. Berlekamp, John H. Conway, Richard K. Guy, *Winning Ways for your Mathematical Plays*, volume 2. Academic Press, 1982, chapitre 25.

[Berners-Lee 90] Tim Berners-Lee, « Information Management : A Proposal », CERN, mars 1989, mai 1990.

[Berners-Lee 90] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, Bernd Pollermann, « World-Wide Web: The Information Universe », *Electronic Networking*, volume 1, numéro 2, 1992, pp. 74-82.

[Bollen 96] Johan Bollen, Francis Heylighen, « Algorithms for the Self-Organization of Distributed, Multi-User Networks. Possible Application to the Future World Wide Web », *Cybernetics and Systems '96*, R. Trappl,ed., Austrian Society for Cybernetics, 1996, pp. 911-916.

[Bollen 97] Johan Bollen, Francis Heylighen, « Dynamic and Adaptive Structuring of the World Wide Web Based on User Navigation Patterns », *Proceedings of the Flexible Hypertext Workshop*, Macquarie Computing Reports, Sydney, 1997, pp.13-17.

[Bond 88] Alan H. Bond, Les Gasser, *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Booth 97] Rick Booth, *Inner Loops*, Addison Wesley Developers Press, 1997.

[Bothorel 98] Cécile Bothorel, « Des communautés dynamiques d'agents pour des services de recommandation », *Actes des 6ièmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Pont-à-Mousson, France, novembre 1998.

[Bradshaw 96] Jeffrey Bradshaw, éditeur, *Software Agents*, Menlo Park, California, AAAI Press / The MIT Press, 1996.

[Brooks 85] Rodney A. Brooks, « A Robust Layered Control System for a Mobile Robot », *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, mars 1986, pp. 14-23; aussi MIT AI Memo 864, septembre 1985.

[Brooks 90a] Rodney A. Brooks, « Elephants Don't Play Chess », *Robotics and Autonomous Systems*, volume 6, 1990.

[Brooks 90b] Rodney A. Brooks, « Challenges for Complete Creature Architectures », *First International Conference on Simulation of Adaptive Behavior*, Paris, France, septembre 1990, pp. 434-443.

[Brooks 91a] Rodney A. Brooks, « Integrated Systems Based on Behaviors », *SIGART Bulletin*, volume 2, numéro 4, août 1991, pp. 46-50.

[Brooks 91b] Rodney A. Brooks, « Intelligence Without Reason », *Proceedings of 12th Int. Joint Conf. on Artificial Intelligence*, Sydney, Australie, août 1991, pp. 569-595.

[Brooks 91c] Rodney A. Brooks, « Intelligence Without Representation », *Artificial Intelligence Journal*, numéro 47, 1991, pp. 139-159.

[Brooks 91d] Rodney A. Brooks, « New Approaches to Robotics », *Science*, volume 253, septembre 1991, pp. 1227-1232.

[Brooks 94] Rodney A. Brooks, L.A. Stein, « Building Brains for Bodies », *Autonomous Robots*, volume 1, numéro 1, novembre 1994, pp. 7-25.

[Brooks 97a] Rodney A. Brooks, « From Earwigs to Humans », *Robotics and Autonomous Systems*, volume 20, numéros 2 - 4, juin 1997, pp. 291-304.

[Brooks 97b] Rodney A. Brooks, « The Cog Project », *Journal of the Robotics Society of Japan*, volume 15, numéro 7, octobre 1997, pp. 968-970.

[Brooks 98] Rodney A. Brooks, C. Breazeal (Ferrell), R. Irie, C. Kemp, M. Marjanovic, B. Scassellat, M. Williamson, « Alternate Essences of Intelligence », *AAAI-98. à apparaître*

[Brown 95] Carol Brown, Les Gasser, Daniel E. O'Leary, Alan Sangster, « AI on the WWW, Supply and Demand Agents », *IEEE Expert/Intelligent Systems & Their Applications* volume 10, numéro 4, août 1995.

[Camps 98] Valérie Camps, *Vers une théorie de l'auto-organisation dans les Systèmes Multi-Agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie*, thèse de doctorat, Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier de Toulouse, Toulouse, France, 1998.

[Chang 97] D. Chang; S. Covaci, « The OMG Mobile Agent Facility », *Proceedings of the First International Workshop on Mobile Agents, MA'97*, Berlin, Allemagne, 7-8 avril, 1997.

[Chappell 96] David Chappell, *Au cœur de ActiveX et OLE*, version française de *Understanding ActiveX et OLE*, traduit de l'anglais par Pierre Brandeis et François Leroy, Microsoft Press, 1996.

[Chavez 96] Anthony Chavez, Pattie Maes, « Kasbah: An Agent Marketplace for Buying and Selling Goods », *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, avril 1996.

[Chen 96] Hsinchun Chen, Andrea Houston, Jay Nunamaker, Jerome Yen, « Toward Intelligent Meeting Agents », *Computer*, volume 29, numéro 8, août 1996, pp. 62-70.

[Cheong 96] Fah-Chun Cheong, *Internet Agents: Spiders, Wanderers, Brokers and 'Bots*, New Riders Publishing, 1996.

[Chess 95] David Chess et al., « Itinerant Agents for Mobile Computing », IBM Research Report RC 20010 (03/27/95), IBM Research Division, 1995.

[Chess 97] D.M. Chess; C.G. Harrison, A. Kershenbaum, « Mobile Agents: Are They a Good Idea? » *Mobile object systems: toward the programmable Internet*, Springer-Verlag, Lecture Notes in Computer Science No. 1222, 1997, pp. 25-45.

[Cockayne 97] William R. Cockayne, Michael Zyda, *Mobile Agents*, Manning Assoc, 1997.

[Collinot 96] Anne Collinot, L. Ploix, Alexis Drogoul, « Application de la méthode Cassiopée à l'organisation d'une équipe de robots », Jean-Pierre Müller, Joël Quinqueton, éditeurs, *Actes des 4èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, AFCET, AFIA, Hermes, 1996.

[Cormen 94] Thomas Cormen, Charles Leiserson, Ronald Rivest, *Introduction à l'algorithmique*, Dunod, Paris, France, 1994.

[CTI 96] Francis Van Aeken, Yves Demazeau, Jaroslaw Kozlak, Pierre Ferrent, rapport semestriel I de la marché CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -LEIBNIZ, août 1996.

[CTI 97a] Francis Van Aeken, Yves Demazeau, Alexandre Ribeiro, rapport semestriel II de la marché CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -LEIBNIZ, février 1997.

[CTI 97b] Francis Van Aeken, Yves Demazeau, rapport semestriel III de la marché CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -LEIBNIZ, août 1997.

[CTI 98a] Francis Van Aeken, Yves Demazeau, Daniel Genovese, rapport semestriel IV de la marché CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -LEIBNIZ, février 1998.

[CTI 98b] Francis Van Aeken, Yves Demazeau, Daniel Genovese, rapport semestriel V de la marché CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -LEIBNIZ, août 1998.

[CTI 99] Francis Van Aeken, Yves Demazeau, Daniel Genovese, rapport semestriel VI de la marché CTI N96 1B 06 entre France Télécom - CNET et INPG CNRS -LEIBNIZ, février 1999.

[Chwey 98] Crystal Chwey, « Autonomy in Space », *IEEE Intelligent Systems*, volume 13, numéro 5, septembre - octobre 1998, pp. 36-44.

[Crevier 93] Daniel Crevier, AI: *The Tumultuous History of the Search for Artificial Intelligence*, Harper-Collins Publishers, Inc., 1993.

[da Rocha Costa 93] Antônio Carlos da Rocha Costa, J.M.V. de Castilho, D.M. Claudio, « Functional Processes and Functional Roles in Societies of Computing Agents », *Proceedings of the 10th Brazilian Symposium on Artificial Intelligence*, Porto Alegre, Brésil, 1993.

[da Rocha Costa 94] Antônio Carlos da Rocha Costa, J.F. Hübner, R.H. Bordini, « On Entering an Open Society », *Proceedings of the 11th Brazilian Symposium on Artificial Intelligence*, Fortaleza, Brésil, octobre 1994, pp. 535-546.

[da Rocha Costa 96] Antônio Carlos da Rocha Costa, Yves Demazeau, « Towards a Formal Model of Multi-Agent Systems with Dynamic Organizations », *Proceedings of the 2nd International Conference on Multi-Agent Systems*, Kyoto, MIT Press, USA, 1996, pp. 431-431.

[Dawkins 76] Richard Dawkins, *The Selfish Gene*, Oxford University Press, New York, 1976.

[Demazeau 90] Yves Demazeau, Jean-Pierre Müller, éditeurs, *Decentralized Artificial Intelligence*, Elsevier North-Holland, 1990.

[Demazeau 91] Yves Demazeau, Jean-Pierre Müller, éditeurs, *Decentralized Artificial Intelligence 2*, Elsevier North-Holland, 1991.

[Demazeau 95] Yves Demazeau, « From Interactions to Collective Behaviour in Agent-Based Systems », *Proceedings of the 1st European Conference on Cognitive Science*, Saint Malo, France, 1995.

[Demazeau 96] Yves Demazeau & A. C. Rocha Costa, « Populations and Organisations in Open Multi-Agent Systems », *Proceedings of the First National Conference on Parallel and Distributed Artificial Intelligence*, Hyderabad, India, July 1996.

[Demazeau 97] Yves Demazeau, « Steps Toward Multi-Agent Oriented Programming », *First International Workshop on Multi-Agent Systems, IWMAS'97*, Boston, 1997.

[Demazeau 98] Yves Demazeau, éditeur, *Proceedings of the Third International Conference on Multi-Agent Systems*, IEEE Computer Society, 1998.

[Dennett 87] Daniel C. Dennett, The Intentional Stance, The MIT Press, Cambridge, MA, 1987.

[Dennett 92] Daniel C. Dennett, *Consciousness Explained*, Little, Brown, Waltham, Mass., 1992.

[de Rosnay 70] Joël de Rosnay, *The Macroscope*, Harper & Row, New York, 1979.

[Di Battista 94] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, Ioannis Tollis, « Annotated Bibliography on Graph Drawing Algorithms », *Computational Geometry: Theory and Applications*, 4, 1994, pp. 235-282.

[Doyle 98] Richard J. Doyle, « Guest Editor's Introduction: Spacecraft Autonomy and the Missions of Exploration », *IEEE Intelligent Systems*, volume 13, numéro 5, septembre - octobre 1998, pp. 36-44.

[Dreyfus 65] Hubert Dreyfus, « Alchemy and Artificial Intelligence », *RAND*, Paper P-3244, décembre 1965.

[Dreyfus 93] Hubert Dreyfus, *What Computers Still Can't Do: a Critique of Artificial Reason*, the MIT Press: Cambridge, Massachusetts and London, England, 1993. Edition révisée de *What Computers Can't Do*, 1979, 1972.

[Drogoul 93] Alexis Drogoul, *De la simulation Multi-Agents à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les Systèmes Multi-Agents*, thèse de doctorat, Université Paris VI, Paris, 1993.

[Duda 73] Richard O. Duda, Peter E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, 1973.

[Erman 80] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy, « The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty », *ACM Computing Surveys*, 12(2), juin 1980, pp. 213-253.

[Etzioni 93] Oren Etzioni, « Intelligence Without Robots », AI Magazine 14, 1993.

[Etzioni 94] Oren Etzioni, Daniel Weld, « A Softbot-Based Interface to the Internet », *Communications of the ACM*, volume 37, numéro 7, ACM Press, July 1994, pp. 72-76.

[Etzioni 95] Oren Etzioni, Daniel S. Weld, « Intelligent Agents on the Internet: Fact, Fiction, and Forecast », *IEEE Expert/Intelligent Systems & Their Applications*, volume 10, numéro 4, août 1995, p. 44.

[Fell 94] Lloyd Fell, David Russell, Alan Stewart, éditeurs., *Seized by Agreement, Swamped by Understanding*, Sydney: Hawkesbury Printing - University of Western Sydney, 1994.

[Ferber 92] Jacques Ferber, Alexis Drogoul, « Using Reactive Multi-Agent Systems in Simulation and Problem Solving », Les Gasser, Nicholas M. Avouris, éditeurs, *Distributed Artificial Intelligence : Theory and Practice*, Kluwer Academic Publishers, 1992.

[Ferber 95] Jacques Ferber, *Les systèmes multi-agents, vers une intelligence collective*, InterEditions, 1995.

[Ferrand 94] Nils Ferrand, A. Labbi, A. Giacometti, B. Amy, Yves Demazeau, « Entre Systèmes Multi-Agents Réactifs et Réseaux d'Automates », 3ème Journée Nationale du PRC-IA sur les Systèmes Multi-Agents, PRC-IA, Paris, France, décembre 1994.

[Filman 98] Robert E. Filman, Sangam Pant, « Searching the Internet », *IEEE Internet Computing*, volume 2, numéro 4, juillet - août 1998, pp. 21-23.

[Finin 94] Tim Finin, Rich Fritzson, Don McKay, Robin McEntire, « KQML as an agent communication language », *Proceedings of the Third International Conference on Information and Knowledge Managment*, ACM Press, 1994.

[Foner 95] Leonard N. Foner, « Clustering and Information Sharing in an Ecology of Cooperating Agents -or- How to Gossip Without Spilling the Beans », *Proceedings of the 1995 Conference on Computers, Freedom, and Privacy*, Burlingame, California, mars 1995.

[Foner 96a] Leonard N. Foner, « A Multi-Agent Referral System for Matchmaking », *The First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology*, London, UK, avril, 1996.

[Foner 96b] Leonard N. Foner, « A Security Architecture for Multi-Agent Matchmaking », *The Second International Conference on Multi-Agent Systems*, Keihanna Plaza, Kansai Science City, Japan, décembre, 1996.

[Foner 97a] Leonard N. Foner, « What's an Agent, Anyway? A Sociological Case Study », *The Proceedings of the First International Conference on Autonomous Agents (AA '97)*, Marina del Rey, California, février 5-8, 1997.

[Foner 97b] Leonard N. Foner, « Yenta: A Multi-Agent, Referral Based Matchmaking System », *The Proceedings of the First International Conference on Autonomous Agents (AA '97)*, Marina del Rey, California, février 5-8, 1997.

[Franklin 96] Stan Franklin, Art Grasser, « Is It an Agent, or Just a Program? A Taxonomy for Autonomous Agents », *Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages*, ECAI-96, Budapest, Hongrie, août 12-13, 1996.

[Fränti 97] Pasi Fränti, Juha Kivijärvi, Timo Kaukoranta, Olli Nevalainen, « Genetic Algorithms for Large-Scale Clustering Problems », *The Computer Journal*, volume 40, numéro 9, 1997.

[Fünfrocken 98] S. Fünfrocken, « Transparent Migration of Java-Based Mobile Agents: Capturing and Reestablishing the State of Java Programs », *Proceedings of the Second International Workshop on Mobile Agents 98 (MA'98)*, Stuttgart, Allemagne, 9-11 septembre, 1998.

[Gaffin 94] A. Gaffin, J. Heitkötter, *EFF's Guide to the Internet*, the Electronic Frontier Foundation, 1994.

[Gardner 83] Martin Gardner, *Wheels, Life, and other Mathematical Amusements*, W. H. Freeman, New York, 1983, chapitres 20-22.

[Genesereth 94] Michael R. Genesereth, Steven P. Ketchpel, « Software Agents », *Communications of the ACM*, volume 37, numéro 7, ACM Press, July 1994, pp. 48-53, 147.

[Genovese 97] Daniel Genovese, *Les agents mobiles pour le World Wide Web : étude comparative*, rapport du probatoire au Conservatoire National des Arts et Métiers, juin 1997.

[Genovese 99] Daniel Genovese, *Réalisation d'un système opérationnel pour la recherche et la classification automatique d'informations sur l'Internet*, mémoire du diplôme d'ingénieur CNAM, Conservatoire National des Arts et Métiers, mars 1999.

[Geomed 95] Geomed Consortium. Final Report. Technical Report, Feasability Study CEC-Telematics, Information Engineering Project, IE-174, Sankt Augustin, Germany, July 1995.

[Gordon 93] A. D. Gordon, Classification: Methods for the Exploratory Analysis of Multivariate Data, Chapman and Hall, 1993.

[Gosling 95] James Gosling & Henry McGilton, « The Java Language Environment: A White Paper », rapport technique, Sun Microsystems, 1995.

[Gray 95] Robert S. Gray, « Agent Tcl: A transportable agent system », James Mayfield and Tim Finin, eds., *Proceedings of the CIKM Workshop on Intelligent Information Agents, Fourth International Conference on Information and Knowledge Management (CIKM 95)*, Baltimore, Maryland, décembre 1995.

[Gudivada 97] Venkat N. Gudivada, Vijay V. Raghavan, William I. Grosky, Rajesh Kasanagottu, « Information Retrieval on the World Wide Web », *IEEE Internet Computing*, volume 1, numéro 5, septembre - octobre 1997, pp. 58-68.

[Gutknecht 98] Olivier Gutknecht, Jacques Ferber, « Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents », *Actes des 6ièmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Pont-à-Mousson, France, novembre 1998.

[Guttman 98] Robert H. Guttman, Pattie Maes, « Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce », *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98)*, ICMAS'98, Paris, France, 3-8 juillet, 1998.

[Harnad 90] Stevan Harnad, « The Symbol Grounding Problem », *Physica D* 42, 1990. Publié dans : Stephanie Forrest, éditeur, *Emergent Computation*, Bradford/MIT Press, Cambridge, MA, 1991, pp. 335-346.

[Harnad 95] Stevan Harnad, « Levels of Functional Equivalence in Reverse Bioengineering », *Artificial Life*, ed. Christopher G. Langton, MIT Press, Cambridge, MA, 1995, pp. 293-301.

[Harrison 95] Colin G.Harrison, David M. Chess, Aaron Kershenbaum, « Mobile Agents: Are they a good idea? », IBM Research Report, IBM Research Divison, mars 1995.

[Hearst 98] Marti A. Hearst, « Information Integration », *IEEE Intelligent Systems*, volume 13, numéro 5, septembre - octobre 1998, p. 12.

[Hewitt 77] Carl Hewitt, « Viewing Control Structures as Patterns of Passing Messages », *Artificial Intelligence* 8(3), 1977, pp. 323-364.

[Hewitt 90] Carl Hewitt, C. Manning, J. Inma, Gul Agha, eds., *Towards Open Information System Science*, MIT Press, Cambridge, MA 1990.

[Hewitt 91] Carl Hewitt, « Open Information Systems Semantics for Distributed Artificial Intelligence », *Artificial Intelligence* 47, 1991, Elsevier, pp. 79-106.

[Heylighen 96] Francis Heylighen, Johan Bollen, « The World-Wide Web as a Super-Brain: from Metaphor to Model », R. Trappl, ed., *Cybernetics and Systems '96*, Austrian Society for Cybernetic Studies, p. 917.

[Holland 75] John Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.

[Howard 96] David M. Howard, James Angus, *Acoustics and Psychoacoustics*, Focal Press, Oxford, 1996.

[Huberman 88] B. A. Huberman, editor, *The Ecology of Computation*, Elsevier Science Publishers B.V., 1988.

[Huhns 97] Michael N. Huhns, Munindar P. Singh, « Mobile Agents », *IEEE Internet Computing*, volume 1, numéro 3, mai - juin 1997, pp. 80-82.

[Huhns 98] Michael N. Huhns, Munindar P. Singh, Les Gasser, éditeurs, *Readings in Agents*, Morgan Kaufman Publishers, 1998.

[Hurst 97] Leon Hurst, Pádraig Cunningham, Fergal Somers, « Mobile Agents - Smart Messages », *Proceedings of the First International Workshop on Mobile Agents, MA'97*, Berlin, Allemagne, 7-8 avril, 1997.

[Ishida 97] Toru Ishida, « Towards Community Ware », *Proceedings of PAAM'97*, The Practical Application Co. Ltd., 1997.

[Ishida 98a] Toru Ishida, ed., *Community Computing, Collaboration over Global Information Networks*, John Wiley & Sons, 1998. Catalogué également sous le titre *Community Ware: Concepts and Practice.*

[Ishida 98b] Toru Ishida, « Supporting Social Events by Mobile Computing », *Proceedings of the 1st International Workshop on Agents in Community Ware*, ICMAS'98, Paris., 1998.

[Jennings 98a] Nicholas R. Jennings, Michael J. Wooldridge, éditeurs, *Agent Technology: Foundations, Applications, and Markets*, Springer Verlag, 1998.

[Jennings 98b] Nicholas R. Jennings, Katia Sycara, Michael Georgeff, éditeurs, *Autonomous Agents and Multi-Agent Systems*, volume 1, Kluwer Academic Publishers, 1998.

[Karjoth 97] Günter Karjoth, Danny B. Lange, Mitsuru Oshima, « A Security Model for Aglets », *IEEE Internet Computing*, volume 1, numéro 4, juillet - août 1997, pp. 68-77.

[Kiniry 97] Joseph Kiniry, Daniel Zimmerman, « A Hands-On Look at Java Mobile Agents », *IEEE Internet Computing*, juillet - août 1997.

[Koch 97] Christof Koch, « Computation and the Single Neuron », *Nature* 385: 207-211, 1997.

[Koning 95] Jean-Luc Koning, Michel Occello, Nils Ferrand, Yves Demazeau, Francis Van Aeken, « A Multi-Agent Approach for Mediation Support on the Net », *1st International Conference on Distributed Intelligent and Multi-Agent Systems*, novembre 1995, Krakovie, Pologne.

[Kotay 94] Keith Kotay, David Kotz, « Transportable Agents », *Proceedings of the CIKM Workshop on Intelligent Information Agents, 3rd International Conference on Information and Knowledge Management, CIKM 94*, Gaithersbury, Maryland, décembre 1994.

[Kotz 97] David Kotz, Robert Gray, Saurab Nog, Daniela Rus, Sumit Chawla, George Cybenko, « AGENT TCL : Targeting the Needs of Mobile Computers », *IEEE Internet Computing*, volume 1, numéro 4, juillet - août 1997, pp. 58-67.

[Krieger 98] David Krieger, Richard M. Adler, « The Emergence of Distributed Component Platforms », *Computer*, volume 31, numéro 3, mars 1998, pp. 43-53.

[Lance 67] G.N. Lance, W.T. Williams, « A General Theory of Classificatory Sorting Strategies. 1 : Hierarchical Systems », *Computer Journal*, 9, 1967, pp. 373-380.

[Lange 98a] Danny B. Lange, Mitsuru Oshima, IBM Research, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, 1998.

[Lange 98b] Danny B. Lange, « Present and Future Trends of Mobile Agent Technology », *Proceedings of the Second International Workshop on Mobile Agents 98 (MA'98)*, Stuttgart, Allemagne, 9-11 septembre, 1998.

[Langton 88] Christopher G. Langton (editor), *1st Conference on Artificial Life*, Addison Wesley, 1988.

[Langton 89] Christopher G. Langton (editor), *Artificial Life*, Santa Fe Institute, Studies in the Sciences of Complexity, Proceedings Volume VI, Addison Wesley, 1989.

[Lashkari 94] Yezdi Lashkari, Max Metral, Pattie Maes, « Collaborative Interface Agents », *Proceedings of AAAI '94 Conference*, Seattle, Washington, août 1994.

[Lassila 98] Ora Lassila, « Web Metadata : A Matter of Semantics », *IEEE Internet Computing*, volume 2, numéro 4, juillet - août 1998, pp. 30-37.

[Lesser 83] Victor R. Lesser, Daniel D. Corkill, « The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks », *AI Magazine*, 4(3), 1983, pp. 15-33.

[Letelier 97] Juan-Carlos Letelier, Fernando Leniz, Francisco Bascuñan, **«** Pitfalls, Risks and Challenges in Teaching Biology of Cognition », *Biology, Language, Cognition and Society: An International Symposium on Autopoiesis*, Belo Horizonte, Brazil, November 18-21, 1997.

[Lieberman 95] Henry Lieberman, « Letizia: An Agent That Assists Web Browsing », *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, Montreal, Canada, août 1995.

[Lieberman 97] Henry Lieberman, « Autonomous Interface Agents », *ACM Conference on Human-Computer Interface [CHI-97]*, Atlanta, mars 1997.

[MacDonald 98] Josh MacDonald, Guy Blelloch, « Clustering », *Algorithms in the Real World*, lecture notes, CS294-3, Carnegie Mellon University, Pittsburgh, 1998.

[Maes 90] Pattie Maes, Rodney Brooks, « Learning to Coordinate Behaviors », *AAAI*, Boston, MA, août 1990, pp. 796-802.

[Maes 94] Pattie Maes, « Agents that Reduce Work and Information Overload », *Communications of the ACM*, volume 37, numéro 7, ACM Press, juillet 1994, pp. 31-40, 146.

[Maes 95] Pattie Maes, « Intelligent Software », *Scientific American*, volume 273, numéro 3, septembre 1995, pp. 84-86.

[Maes 97] Pattie Maes (interview), « Pattie Maes on Software Agents: Humanizing the Global Computer », *IEEE Internet Computing*, volume 1, numéro 4, juillet - août 1997, pp. 10-19.

[Marcenac 97] Pierre Marcenac, « Modélisation de systèmes complexes par agents », Technique et Science Informatiques, 16-8, octobre 1997, pp. 1013-1037.

[MARCIA 96] Groupe MARCIA, « Auto-organisation := évolution de structure(s) », *Actes de la 4ème Journée Nationale du PRC-IA sur les Systèmes Multi-Agents*, PRC-IA, Toulouse, 1996. pp. 139-152.

[Marr 82] David Marr, *Vision*, W.H. Freeman and Company, New York, NY, 1982.

[Maturana 73] Humberto R. Maturana, Francisco J. Varela, « Autopoiesis : the Organization of the Living », article de 1973, republié dans Humberto R. Maturana, Francisco J. Varela, *Autopoiesis and Cognition*, 1980, pp. 63-134.

[Maturana 75] Humberto R. Maturana, « The organization of the living: A theory of the living organization », *International Journal of Man-Machine Studies*, volume 7, 1975, pp. 313-332.

[Maturana 78] Humberto R. Maturana, « Biology of language: The epistemology of reality », chapitre 2 dans *Psychology and Biology of Language and Thought: Essays in Honor of Eric Lenneberg*, G.A. Miller, E. Lenneberg, eds., New York: Academic Press, 1978, pp. 27-63.

[Maturana 80] Humberto R. Maturana, Francisco J. Varela, *Autopoiesis and Cognition: the Realization of the Living*, Boston Studies in the Philosophy of Science, volume 42, 1980.

[Maturana 87] Humberto R. Maturana, Francisco J. Varela, traduit par Robert Paolucci**,** *The Tree of Knowledge; The biological Roots of Human Understanding***,** Boston, Mass., Shambhala Publications, 1987.

[Maturana 88] Humberto R. Maturana, « Ontology of Observing. The Biological Foundations Of Self Consciousness And the Physical Domain Of Existence », *Conference Workbook: Texts in Cybernetics*, American Society For Cybernetics Conference, Felton, CA, 18-23 octobre, 1988.

[Maturana 88] Humberto R. Maturana, « Reality: The Search for Objectivity or the Quest for a Compelling Argument », *The Irish Journal of Psychology* 9, 1, 1988, pp. 25-82.

[McCarthy 55] John McCarthy, Marvin L. Minsky, N. Rochester, C. E. Shannon, « A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence », 1955.

[Merlat 97a] Walter Merlat, Claude Seyrat, Jacques Ferber, « Mobile Agents for Dynamic Organisations : The Conversational-Agent Paradigm », *Proceedings of the 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, MAAMAW'97*, 1997. Présenté comme poster.

[Merlat 97b] Walter Merlat, Claude Seyrat, « JavaNetAgents : Une plate-forme d'exécution d'agents mobiles pour le développement de Systèmes Multi-Agents sur Internet », *Actes des 5èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, AFCET, AFIA, Quinqueton, Thomas, Trousse, eds., 1997.

[Merlat 98] Walter Merlat, *Adaptation dynamique de l'organisation dans les Systèmes Multi-Agents. Application à la conception des Systèmes Coopératifs Distribués et Ouverts*, thèse de doctorat, Université Paris VI, 1998.

[Milojicic 98] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, J. White , « MASIF: The OMG Mobile Agent System Interoperability Facility », *Proceedings of the Second International Workshop on Mobile Agents 98 (MA'98)*, Stuttgart, Allemagne, 9-11 septembre, 1998.

[Minar 98] Nelson Minar, « Designing an Ecology of Distributed Agents », M.S. Thesis, Massachusetts Institute of Technology, 1998.

[Minsky 54] Marvin Minsky, *Neural Nets and the Brain Model Problem*, thèse de doctorat, Princeton University, 1954.

[Minsky 69] Marvin Minsky, Seymour A. Papert, *Perceptrons*, MIT Press, MIT Press, Cambridge, Massachusetts, 1969.

[Minsky 86] Marvin Minsky, *The Society of Mind*, Simon & Schuster, New York, 1986.

[Moravec 90] Hans Moravec, *Mind Children: The Future of Robot and Human Intelligence*, Harvard University Press, 1990.

[Moukas 96] Alexandros G. Moukas, « Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem », *Proceedings of the Conference on Practical Application of Intelligent Agents and Multi-Agent Technology*, London, 1996.

[Myhrvold 97] Nathan Myhrvold, « The Next Fifty Years of Software », *Slides of the ACM Conference on the Next Fifty Years of Computing*, San Jose, California, 1-5 mars 1997.

[Nelson 67] Ted Nelson, « Getting It Out of Our System », George Schecter, éditeur, *Information Retrieval: A Critical View*, Thompson Books, Washington D.C., 1967, pp. 191-210.

[Newell 80] Allen Newell, « Physical Symbol Systems », *Cognitive Science* 4 :135-83, 1980.

[Occello 96] Michel Occello, Yves Demazeau, « Une approche du temps réel dans la conception d'agents », *Actes des 4èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, AFCET, AFIA, Müller, Quinqueton, eds., Hermes, 1996 pp. 101-111.

[Occello 97a] Michel Occello, Yves Demazeau, « CELLO : A Model of Agent with Real Time Constraints », *1st International Conference on Autonomous Agents*, Marina del Rey, 1997.

[Occello 97b] Michel Occello, Yves Demazeau, « Vers une approche de conception et de description récursive en univers multi-agents », *Actes des 5èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents*, AFCET, AFIA, Quinqueton, Thomas, Trousse, eds., 1997, pp. 143-157.

[O'Hare 96] Greg M.P. O'Hare & Nick R. Jennings, *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, 1996.

[Olson 94] Clark F. Olson, « Parallel Algorithms for Hierarchical Clustering », rapport technique UCB//CSD-94-786, Computer Science Division, University of California at Berkeley, Berkeley, Californie, 1994.

[Ousterhout 88] John K. Ousterhout, A. R. Cherenson, Fred Douglis, Michael N. Nelson, Brent B. Welch, « The Sprite Network Operating System », *Computer*, 21(2), février 1988, pp. 23-36.

[Perram 97] John W. Perram, Yves Demazeau, « A Multi-Agent Architecture for Distributed Constrained Optimization and Control », *6th Scandinavian Conference on Artificial Intelligence*, Helsinki, 1997.

[Perret 96a] Stéphane Perret, Andrzej Duda, « Design and Implementation of MAP: A System for Mobile Assistant Programming », *Proceedings of the IEEE International Conference on Parallel and Distributed Systems*, Tokyo, juin 1996.

[Perret 96b] Stéphane Perret, Andrzej Duda, « MAP: Mobile Assistant Programming for Large Scale Communication Networks », *Proceedings of the IEEE International Communications Conference 96*, Dallas, juin 1996.

[Perret 96c] Stéphane Perret, Andrzej Duda, « Mobile Assistant Programming for Efficient Information Access on the WWW », *Computer Networks and ISDN Systems (Proceedings of the 5th International World Wide Web Conference)*, 1996, pp. 1373-1383.

[Petrie 96] Charles J. Petrie, « Agent-Based Engineering, the Web, and Intelligence », *IEEE Expert/Intelligent Systems & Their Applications*, volume 11, numéro 6, décembre 1996, pp. 24-29.

[Prosise 96] Jeff Prosise, *MFC Programmation sous Windows 95*, version française de *Programming Windows 95 with MFC*, adapté de l'américain par Georges-Louis Kocher, Microsoft Press, 1996.

[Rao 91] Anand S. Rao, Michael P. Georgeff, « Modeling Rational Agents Within a BDI-Architecture », Richard Fikes, Erik Sandewall, éditeurs, *Proceedings of Knowledge Representation and Reasoning, KR&R'91*, Morgan Kaufmann Publishers, San Mateo, CA, pp. 473-484.

[Rasmussen 98] Robert Rasmussen, « What Does Autonomy Capability Really Mean for JPL and NASA? A Talk with Robert Rasmussen », interview, *IEEE Intelligent Systems*, volume 13, numéro 5, septembre - octobre 1998, pp. 76-77.

[Reddy 96] Raj Reddy, « The Challenge of Artificial Intelligence », *Computer*, volume 29, numéro 10, octobre 1996, pp. 86-98.

[Rhodes 96] Bradley J. Rhodes, « Remembrance Agent: A Continuously Running Automated Information Retrieval System », *Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology*, Londres, avril 1996, pp. 487-495.

[Roederer 94] Juan G. Roederer, *The Physics and Psychophysics of Music*, Springer-Verlag, New York, 1994.

[Rothermel 97] Kurt Rothermel, Radu Popescu-Zeletin, eds., *Proceedings of the First International Workshop on Mobile Agents, MA'97*, Lecture Notes in Computer Science, 1219, Springer Verlag.1997.

[Rothermel 98] Kurt Rothermel, Fritz Hohl, eds., *Proceedings of the Second International Workshop on Mobile Agents, MA'98*, Springer-Verlag, Allemagne, 1998.

[Russell 03] Bertrand Russell, *Principles of Mathematics*, Cambridge: Cambridge University Press, 1903.]

[Russell 08] Bertrand Russell, « Mathematical Logic as Based on the Theory of Types », *American Journal of Mathematics*, 30, 1908, pp. 222-262. Aussi dans Bertrand Russell, *Logic and Knowledge*, London: Allen & Unwin, 1956, pp. 59-102, et dans Jean van Heijenoort, *From Frege to Gödel*, Cambridge, Mass.: Harvard University Press, 1967, 152-182.]

[Searle 69] John R. Searle, *Speech Acts*, Cambridge University Press, 1969.

[Searle 80] John R. Searle, « Minds, Brains and Programs », *Behavior and Brains Sciences*, 1980. Egalement dans Margaret A. Boden, editeur, *The Philosophy of Artificial Intelligence*, Oxford University Press, New York, 1990, pp. 67-88.

[Shannon 38] Claude E. Shannon, *A Symbolic Analysis of Relay and Switching Circuits*, thèse de doctorat, Massachusetts Institute of Technology, 1938.

[Shannon 48] Claude E. Shannon, « A Mathematical Theory of Communication », *Bell System Technical Journal*, volume 27, juillet et octobre, 1948, pp. 379-423 et 623-656.

[Sian 91] S. Sian, « Adaptation Based on Cooperative Learning in Multi-Agent Systems », *Decentralized AI 2*, eds. Y. Demazeau and J.-P. Muller, Elsevier Science Publishers, 1991.

[Singh 97] Munindar P. Singh, Michael N. Huhns, « Internet-Based Agents: Applications and Infrastructure », *IEEE Internet Computing*, volume 1, numéro 4, juillet - août 1997, pp. 8-9.

[So 96] Young-pa So, Edmund H. Durfee, « Designing Tree-Structured Organizations for Computational Agents », *Computational and Mathematical Organization Theory*, 2(3), 1996, pp. 219-246.

[Steels 91] Luc Steels, « Towards a Theory of Emergent Functionality », *From Animals to Animats*, eds. J.-A. Meyer and S. Wilson, MIT Press, Cambridge, MA, 1991, p. 451.

[Steels 94] Luc Steels, « The Artificial Life Roots of Artificial Intelligence », *Artificial Life Journal* volume 1, numéro 1, MIT Press, Cambridge, 1994.

[Steels 95] Luc Steels, Rodney Brooks, eds., *The Artificial Life Route to Artificial Intelligence: Building Embodied Situated Agents*, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1995.

[Steels 96a] Luc Steels, « The origins of intelligence », *Proceedings of the Carlo Erba Foundation Conference on Artificial Life*, Fondazione Carlo Erba, Milano, 1996.

[Steels 96b] Luc Steels, « Perceptually grounded meaning creation », *Proceedings of ICMAS-96*, Tokoro, M., ed., Kyoto 1996. Calif, AAAI Press.

[Stein 94] Lynn A. Stein, « Intelligence and Reason: A Response to Etzioni », Letter to the Editor, *AI Magazine* volume 15, numéro 2, été 1994, pp. 11-12.

[Stinckwich 94] Serge Stinckwich, *Modèles organisationnels et réflexifs des architectures à objets concurrents. Implémentation en Smalltalk-80*, Thèse de Doctorat, ENS Lyon, 1994.

[Stroustrup 97] Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley, 1997.

[Tanaka 95] Y. Tanaka, « Meme Media and its World-Wide Pool for the Exchange and Evolution of Knowledge », *Proceedings of the 20th International Conference on Information Technologies and Programming*, Peter Barnev, ed., Plovdiv, Bulgarie, 1995.

[Taylor 94] Charles Taylor, *Exploring Music*, IOP Publishing Ltd, 1994.

[Taylor 95] Charles Taylor, David Jefferson, « Artificial Life as a Tool for Biological Inquiry », *Artificial Life*, Christopher G. Langton, ed., MIT Press, 1995.

[Tichy 98] Walter F. Tichy, « Should Computer Scientists Experiment More? », *Computer*, volume 31, numéro 5, mai 1998, pp. 32-40.

[Tokoro 96] Mario Tokoro, editeur, *Proceedings of the Second International Conference on Multi-Agent Systems*, AAAI Press, 1996.

[Turing 50] Alan M. Turing, « Computing Machinery and Intelligence », *Mind* 49, 1950, pp 433-460. Aussi dans *Minds and Machines*, ed. A. Anderson Prentice Hall, Engelwood Cliffs, NJ, 1964.

[Van Aeken 90a] Francis Van Aeken, *Elastic Patterns Go Parallel Distributed*, Licentiaatsverhandeling, Vrije Universiteit Brussel, Bruxelles, Belgique, 1990.

[Van Aeken 90b] Francis Van Aeken, Yves Demazeau, « Processing Elastic Patterns the Massively Parallel Way », *Proceedings of the IAPR Workshop on Machine Vision Applications*, Tokyo, Japan, novembre 28-30, 1990.

[Van Aeken 94] Francis Van Aeken, *The Art of Insanity*, manuscrit non publié, Bruxelles, Belgique, 1994.

[Van Aeken 96a] Francis Van Aeken, « Le Salon, où des agents rencontrent des acteurs », *4ièmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Port Camargue, France, avril 1-3, 1996. (présenté comme poster)

[Van Aeken 96b] Francis Van Aeken, « In Search of a Common Language », *AI Tools Workshop of the 5th International World Wide Web Conference*, Paris, France, mai 6-9, 1996.

[Van Aeken 96c] Francis Van Aeken, Yves Demazeau, « When Agents Talk - How to Maintain Integrity », *IATA Workshop of the 12th European Conference on Artificial Intelligence*, Budapest, Hongrie, août 12-16, 1996.

[Van Aeken 96d] Francis Van Aeken, « Le Salon, Where Agents Meet Actors », *Proc. 8th IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France, novembre 16-19, 1996, p. 432-433.

[Van Aeken 97] Francis Van Aeken, Yves Demazeau, Jaroslaw Kozlak, « Migration, Mobility, and The Palindrome Problem », *Proc. 1st International Workshop on DAI and MAS*, Saint-Petersburg, 1997, p. 23-31. Aussi dans *: Proc. 22th International Conference on Information Technology and Processing , IT&P'97: Multimedia Technology and Systems*, Sofia, Bulgarie, 1997, p. 7-14.

[Van Aeken 98a] Francis Van Aeken, Yves Demazeau, « On the Integration of Multi-Agent System Technology in the Information Society », *Proc. 23th International Conference on Information Technology and Processing , IT&P'98: The Information Society*, Sofia, Bulgarie, 1998.

[Van Aeken 98b] Francis Van Aeken, Yves Demazeau, « Minimal Multi-Agent Systems », *Proc. 3rd International Conference on Multi-Agent Systems*, Paris, France, juillet 1998.

[Van Aeken 98c] Francis Van Aeken, Yves Demazeau, « Les Systèmes Multi-Agents Minimaux », *Actes des 6ièmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Pont-à-Mousson, France, novembre 1998.

[Vanderveken 90] Daniel Vanderveken, *Meaning and Speech Acts*, volume I, Cambridge University Press, New York, 1990.

[Van Dyke 99] Neil W. Van Dyke, Henry Lieberman, Pattie Maes, « Butterfly: A Conversation-Finding Agent for Internet Relay Chat », *Proceedings of 1999 International Conference on Intelligent User Interfaces*, janvier 1999.

[Varela 89] Francisco J. Varela, *Autonomie et connaissance*, Editions du Seuil, Paris, 1989.

[Vigna 98] Giovanni Vigna, editeur, *Mobile Agents and Security*, Springer-Verlag, Allemagne, 1998.

[Vittal 81] J. Vittal, « Active Message Processing: Messages as Messengers », *Computer Message Systems*, North-Holland Publishing Company, 1981.

[von Bertalanffy 68] Ludwig von Bertalanffy, *General Systems Theory, Foundations, Development, Application*, G. Braziller, New York, 1968.

[von Foerster 74] Heinz von Foerster, editeur, *Cybernetics of Cybernetics*, Biological Computer Laboratory, University of Illinois, Urbana, Illinois, 1974.

[Von Neumann 45] John von Neumann, *First Draft of a Report on the EDVAC*, Contract No. W-670-ORD-492, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, 1945, réimprimé dans N. Stern, *From ENIAC to UNIVAC*, Digital Press, Bedford, Mass., 1981,

pp. 177-246, et (partiellement) dans Brian Randell, *Origins of Digital Computers: Selected Papers*, Springer-Verlag, Berlin Heidelberg, 1982, pp. 383-392.

[Walsh 98] Tom Walsh, Noemi Paciorek, David Wong, « Security and Reliability in Concordia », *Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS31)*, Kona, Hawaii, January 6-9, 1998.

[Walter 50] W.G. Walter, « An Imitation of Life », *Scientific American*, 182 (5), 1950, pp. 42-45.

[Walter 51] W.G. Walter, « A Machine That Learns », *Scientific American*, 185 (5), 1951, pp. 60-63.

[Whitaker 95] R. Whitaker, « Self-Organization, Autopoiesis, and Enterprises », *ACM SIGOIS Illuminations*, December 1995.

[White 95] James E. White, « Telescript Technology: An Introduction to the Language », General Magic White Paper GM-M-TSWP3-0495-V1, General Magic, Inc., 420 North Mary Avenue, Sunnyvale, CA 94086, 1995.

[White 96] James E. White, « Telescript Technology: Foundation for the Electronic Marketplace », white paper, General Magic, Sunnyvale, California, 1996. Une autre version de l'article se trouve dans Bradshaw, Jeffrey (ed.) *Software Agents*, Menlo Park, California, AAAI Press / The MIT Press, 1996.

[White 97] James E. White, C. S. Helgeson, and D. A. Steedman, « System and method for distributed computation based upon the movement, execution, and interaction of processes in a network », US Patent 5,603,031, filed 8 filed 1993, issued 11 February 1997.

[Wiener 48] Norbert Wiener, *Cybernetics, or Control and Communication in the Animal and the Machine*, John Wiley & Sons, Inc., 1948.

[Winograd 75] Terry Winograd, « Frame Representations and the Declarative / Procedural Controversy », dans *Representation and Understanding*, D. Bobrow, A. Collins, eds., Academic Press, New York, 1975. Aussi dans Readings in Knowledge Representation, R. Brachman, H. Levesque, eds., pp. 358-370, Morgan Kaufmann, San Francisco, 1985.

[Wong 97] David Wong, Noemi Paciorek, Tom Walsh, Joe DiCelie, Mike Young, Bill Peet, « Concordia: An Infrastructure for Collaborating Mobile Agents », *Proceedings of the First International Workshop on Mobile Agents, MA'97*, Berlin, Allemagne, 7-8 avril, 1997.

[Wooldridge 95] Michael Wooldridge, Nick Jennings, « Intelligent Agents: Theory and Practice », *The Knowledge Engineering Review* 10 (2), 1995, pp. 115-152.

[Yoshida 98] Sen Yoshida et al, « Visualizing Potential Communities : A Multi-Agent Approach », *Proceedings of the 1st International Workshop on Agents in Community Ware, ICMAS'98*, Paris, France, 1998.

[Zelnick 98] Nate Zelnick, « Nifty Technology and Nonconformance: The Web in Crisis », *Computer*, Octobre 1998, pp. 115-116, 119.

Liens vers des pages sur le World Wide Web — nous avons vérifié leur existance le 13 février 1999.

[AltaVista] http://www.altavista.com/

[Amazon] http://amazon.com

[Agent Society] http://www.agent.org/

[Agent Tcl] http://www.cs.dartmouth.edu/~agent/

[aglets] http://www.trl.ibm.co.jp/aglets/

[Ara] http://www.uni-kl.de/AG-Nehmer/Projekte/Ara/index_e.html

[Café] http://cafe.symantec.com/

[CNET] http://www.cnet.fr/

[Concordia] http://www.meitca.com/HSL/Projects/Concordia/

[constructivisme] http://www.univie.ac.at/cognition/constructivism/index.html

[Cyc] http://www.cyc.com

[Dartmouth] http://www-formal.stanford.edu/jmc/history/dartmouth.html

[Deep Blue] http://www.chess.ibm.com/home/html/b.html

[Deep Blue FAQ] http://www.chess.ibm.com/meet/html/d.3.3a.html#ai

[dialectes] http://odur.let.rug.nl/~heeringa/dialectology/dial.htm

[entropie] http://www.math.washington.edu/~hillman/entropy.html

[ERLI] http://www.erli.com/

[Excite] http://www.excite.com/

[Fell] http://bart.northnet.com.au/~pfell/book.html

[FIPA] http://drogo.cselt.stet.it/fipa/

[Firefly] http://www.firefly.com

[graph drawing] http://www.cs.brown.edu/people/rt/gd.html

[HotBot] http://www.hotbot.com/

[HTML 4.0] http://www.w3.org/TR/REC-html40/

[IBM] http://www.networking.ibm.com/iag/iaghome.html

[infoseek] http://www.infoseek.com/

[Ingrid] http://www.ingrid.org/

[Intel] http://www.intel.com/pressroom/kits/processors/quickref.htm

[Java] http://java.sun.com

[JavaBeans] http://java.sun.com/beans/

[Java-To-Go] http://ptolemy.berkeley.edu/dgm/javatools/java-to-go/

[JavaWorld] http://www.javaworld.com/

[Lange] http://ncstrl.informatik.uni-stuttgart.de/ipvr/vs/ws/ma98/presentations/danny.pdf

[Lycos] http://www.lycos.com/

[MA’99] http://www.genmagic.com/asa/

[Magellan] http://magellan.excite.com/

[MAGMA] http://www-leibniz.imag.fr/MAGMA/

[MAP] http://fidji.imag.fr/map.html

[MetaCrawler] http://www.metacrawler.com/

[MetaFind] http://www.metafind.com/

[Mole] http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole.html

[Moore] http://www.intel.com/intel/museum/25anniv/hof/moore.htm

[Observer] http://www.informatik.umu.se/~rwhit/AT.html

[Odyssey] http://www.genmagic.com/technology/odyssey.html

[OMG] http://www.omg.org/

[paradoxe] http://plato.stanford.edu/entries/russell-paradox/

[Principia Cyber.] http://pespmc1.vub.ac.be/

[QuiQuoiOù] http://www.wanadoo.fr/qqo

[software agents] http://agents.www.media.mit.edu/groups/agents/

[SGP] http://www.cogsci.soton.ac.uk/~harnad/Papers/Harnad/harnad90.sgproblem.html

[TACOMA] http://www.tacoma.cs.uit.no/

[Telescript] http://www.genmagic.com/technology/techwhitepaper.html

[Varela] http://shr.stanford.edu/shreview/4-2/text/varela.html

[Verity] http://www.verity.com/

[Visual C++] http://msdn.microsoft.com/visualc/prodinfo/previous/v5/vcawardpr11.asp

[Voyager] http://www.objectspace.com/products/voyager/index.html

[Wanadoo] http://www.wanadoo.fr/

[WebCrawler] http://www.webcrawler.com/

[Wizards] http://www.nw.com/zone/WWW/top.html

[XML] http://www.w3.org/XML/

[Yahoo] http://www.yahoo.com