Proxy d'interface Homme-Machine : apport des algorithmes génétiques pour l'adaptation automatique de la présentation de documents Web

Jérémy Lardon

Thèse

Pour obtenir le grade de

Docteur de l'université Jean Monnet de Saint-Étienne

Discipline : Informatique

Présentée et soutenue publiquement le 29 novembre 2010

par

Jérémy LARDON

Maître ès Sciences

Proxy d'Interface Homme-Machine : Apport des algorithmes génétiques pour l'adaptation automatique de la présentation des documents Web

Directeur de thèse :
Jacques FAYOLLE

Co-Directeur de thèse :
Christophe GRAVIER

Composition du Jury :

Amel Bouzeghoub	Professeur, SAMOVAR UMR CNRS 5157, Télécom SudParis, Institut Télécom	*Rapporteur*
Frédérique Laforest	Maître de Conférences HDR, LIRIS UMR CNRS 5205, INSA de Lyon	*Rapporteur*
Cyril Concolato	Maître de Conférences, LTCI UMR CNRS 5141, Télécom ParisTech, Institut Télécom	
Pierre Maret	Professeur des Universités, LHC UMR CNRS 5516, Université Jean Monnet	
Jacques Fayolle	Maître de Conférences HDR, Université Jean Monnet	*Directeur*
Christophe GRAVIER	Chargé de recherche, Télécom Saint-Etienne	*Co-directeur*

Remerciements

Ce travail a été réalisé au sein de l'équipe SATIn - Laboratoire des *Dispositifs et Instrumentation en Opto-ectronique et Micro-ondes* (DIOM) -Télécom Saint-Etienne - Université Jean MONNET de Saint-Etienne - Université de Lyon, grâce à un financement sur fonds propres de l'école d'ingénieures Télécom Saint-Etienne.

Je tiens en premier lieu à remercier la société LOTIM Télécoms et le Conseil Général de la Loire ayant permis de constituer ce financement.

Je remercie mon directeur de thèse, Jacques Fayolle, pour son enthousiasme, son soutien et la confiance sans faille qu'il m'a accordée durant ces travaux doctoraux.

Je remercie également mon co-directeur de thèse, Christophe Gravier, pour ses conseils et sa disponibilité.

Mes remerciements vont aussi à Amel Bouzeghoub et à Frédérique Laforest qui ont eu la gentilesse de rapporter sur ce manuscrit. Leurs retours et critiques constructives ont permis de donner une consistence à ma plume scientifique.

Je remercie Cyril Concolato et Pierre Maret pour avoir bien voulu faire parti de mon jury.

Je remercie mes collègues et camarades de brainstorming, relecteurs attentionnés, Mikaël, Christophe, Yves-Gaël, Benjamin et Antoine. Beaucoup de bons moments partagés lors de mon passage à Télécom Saint-Etienne.

Je remercie tous mes amis pour m'avoir redonné le moral dans les mauvais jours et pour leur disponibilité.

Enfin, un grand merci à ma famille, pour les relectures et les encouragements.

Résumé

L'informatique pervasive, paradigme fer de lance des services "anytime/anywhere", appelle de plus en plus à des travaux de ré-ingénierie de sites Web. L'approche contemporaine de l'informatique dans les nuages va d'ailleurs générer de nouveaux besoins en ce sens. Aujourd'hui, les sites Web sont toujours pensés pour l'affichage sur un ordinateur traditionnel (comprendre *desktop*). Des versions alternatives sont cependant de plus en plus proposées pour l'accès via des smartphones, ou encore des dispositifs de visionage jadis passifs, comme la télévision (interactive IPTV).

Ces travaux de développement sont souvent relégués à des tâches *ad hoc* dans la gestion de projet du développement d'un site Web. La demande croissante en terme de services numériques Web pervasifs devient alors un enjeu économique dans la société numérique: l'automatisation de ces travaux d'ingénierie serait un facteur d'économie d'échelle certain.

Même si les plates-formes mobiles offrent de plus en plus de mécanismes pour faciliter la consultation et l'interaction avec les contenus Web, tels que les systèmes de zoom, de vignettes (traduction la plus proche du mot anglais *thumbnail*) ou encore de loupe, ceci ne permet pas d'adresser tout type de contenu Web. Ce n'est également pas généralisable en l'état pour les autres plates-formes comme les SetTop Box, permettant l'affichage de documents Web sur un téléviseur, ou encore les dispositifs de consultation en émergence comme les tables interactives, murs numériques, etc. Sans parler des dispositifs qui restent à inventer demain. Il nous paraît donc nécessaire de pouvoir adapter un document Web selon les contraintes induites par la plate-forme de consultation.

La problématique soulevée est la suivante : est-il possible, et si oui comment, d'automatiser la transformation de la présentation de contenus Web pour un dispositif cible non connu à l'avance ?

Pour répondre à cette problèmatique d'adaptation, trois approches se distinguent.

- La première repose sur les solutions *ad hoc* pour adapter la présentation d'un document Web, solution passant difficilement à l'échelle.
- Le transcodage est une transformation au niveau du code du document Web pour satisfaire un autre contexte.
- Au contraire, la ré-ingénierie a pour but d'abstraire la présentation du document

vers un méta-modèle, de transformer au besoin ce modèle pour qu'il réponde mieux au nouveau contexte, puis de décliner l'instance du méta-modèle vers l'architecture cible.

La thèse repose sur un modèle d'adaptation voisin du transcodage portant sur la représentation DOM du document. A l'image du transcodage, nous faisons appel à des techniques appelées transformations dans notre modèle. Les transformations étant multiples, il est nécessaire de déterminer lesquelles doivent être appliquées et dans quel ordre pour chaque contexte cible. Ce séquencement de transformations est appelé composition.

Cette thèse démontre qu'une approche par méta-heuristique permet, dans des limites de fonctionnement identifiées, d'automatiser la ré-ingénierie d'un site Web traditionnel pour un dispositif de consultation non connu par avance. Pour ce faire, nous proposons une architecture couplée à un moteur permettant de conserver la session de navigation de l'utilisateur dans le cas de découpe du contenu Web en écrans successifs.
Les caractéristiques des dispositifs n'étant pas connues par avance, nous proposons également un mécanisme permettant de moissoner passivement les informations du dispositif de consultation, afin de prendre en compte les capacités d'affichage et de navigation dans nos compositions. Nous utilisons une approche basée sur un serveur mandataire de présentation de document Web, ainsi que le paradigme de cookie transdomaine (cross-domain cookie).

Les algorithmes génétiques nous permettent d'approximer le problème d'optimisation de cette composition et son séquencement. En effet, les différentes compositions possibles sont mises en concurrence, croisées, et évaluées, de sorte qu'une solution proche d'un optimum puisse se dégager en un temps fini. Cet algorithme est au cœur du moteur d'adaptation proposé.

La thèse fait état de l'implémentation de ce modèle complet, des résultats obtenus expérimentalement, et propose une interprétation de la performance du système.

Le premier chapitre de ce mémoire est consacré au contexte de notre étude et l'étude bibliographique. Tout d'abord, nous présentons les généralités sur le domaine de l'adaptation automatique de documents Web. Par la suite, nous donnons un aperçu des contributions scientifiques ainsi que des systèmes déjà développés pour l'adaptation automatique. L'étude comparative de ces travaux nous a permis de dégager les pistes de travail de l'ap-

proche proposée dans la thèse

Le second chapitre propose notre modèle et plus particulièrement son découpage architectural. Nous présentons les concepts d'estimation de valeurs caractéristiques et de simulation de transformations permettant de construire l'algorithme génétique au centre de notre moteur d'hypermédia adaptatifs. L'implémentation de notre algorithme génétique y est également explicitée.

Enfin, le chapitre 3 sert à présenter l'observation des résultats obtenus, ainsi que dresser les conclusions liées à notre implémentation. Cette présentation est associée à une étude des conséquences de la variation des paramètres de notre modèle et des ressources computationnelles. De cette analyse, nous soulevons les perspectives qu'offrent nos travaux.

Mots-clés: Hypermédia adaptatifs, Web, DOM, algorithmes génétiques, transcodage, ré-ingénierie.

Table des matières

Table des figures

Liste des tableaux

CHAPITRE I

Problématique et état de l'art

Sommaire

CHAPITRE I

Problématique et état de l'art

Introduction

Ce premier chapitre a pour but de poser les bases de compréhension du travail présenté dans ce manuscrit. Nous expliquons tout d'abord le contexte de l'adaptation dynamique de documents Web. Par la suite, nous donnons les définitions du domaine de recherche. Ceci fait, nous examinons les apports passés et actuels, scientifiques et techniques, à l'adaptation automatique. De cette étude de l'existant, nous mettrons en exergue les limitations des systèmes existants dédiés à l'adaptation automatique de documents Web. Enfin nous définirons les objectifs de nos travaux doctoraux.

I.1 Enjeux de l'adaptation dynamique de documents Web

I.1.1 Nomadisme et informatique pervasive

L'usage d'Internet s'est fait une place dans nos vies. En effet plus d'un français sur trois s'est déjà connecté à Internet (Médiamétrie, 2009b). Parmi les applications populaires d'Internet, nous retrouvons le World Wide Web (souvent raccourci en « Web »). Il est une source incommensurable d'informations.

Cette utilisation d'Internet est d'autant plus importante qu'avec l'introduction des plates-formes portables de navigation, les utilisateurs peuvent accéder aux contenus Web quand et d'où ils le souhaitent. On parle souvent d'informatique pervasive, du latin "pervadere" (aller de toute part, s'insinuer, se propager, se répandre) pour décrire ce phénomène d'adoption d'Internet. Ce paradigme est autrement connu sous le leitmotiv : "anytime/anywhere". Il n'est plus à prouver que les plates-formes mobiles (téléphones portables, PDAs, smartphones, netbooks) sont largement utilisées pour cette usage (Médiamétrie, 2009a).

A ce mode de consultation nomade du Web, on peut ajouter un média qui est actuellement peu usité : le téléviseur. Certaines Set-Top Box (STB) incorporent en effet un navigateur Web capable d'afficher les pages Web sur un téléviseur. Cette pratique tend à se démocratiser avec l'arrivée de téléviseurs directement reliés à Internet.

Cependant, s'il est possible d'avoir accès aux contenus Web grâce à ces plates-formes, ils restent inadaptés pour celles-ci. En effet, la majorité des pages Web est conçue pour l'affichage sur un ordinateur de bureau. Cela entraîne des difficultés sur les plates-formes citées ci-dessus pour l'affichage des documents Web et l'interaction à travers eux.

Par conséquent, il est possible de considérer ces freins à la consultation des sites Web comme des limitations induites par ces plates-formes. La table I.1 recense les cinq catégories de limitations que nous avons pu extraire ainsi que des exemples pour chacune d'elles.

Pour résumer, l'accès aux informations offert par le Web se fait de plus en plus à travers des terminaux mobiles. Le futur placera-t-il le téléviseur comme le prochain média de consultation des documents hébergés sur Internet ? Plus probablement, le monopole de l'écran/clavier/souris semble terminé au profit partagé de multiples dispositifs hétérogènes pour accéder à l'information. La diversité de ces dispositifs, des utilisateurs et de leur contexte d'usage[1] est une motivation forte quant à la nécessité d'inventer des moyens d'adapter l'information Web.

1. dans notre étude, le contexte est l'ensemble des éléments significatifs de l'environnement utilisateur

Catégories de limitations	Exemples de limitations
Observabilité	– résolution d'affichage – profondeur de couleur
Modalités d'interaction	– Absence de dispositif de pointage (pas de souris, seulement un clavier numérique) – Parcours du document Web d'un élément d'interaction au suivant
Disponibilité	– Vitesse de transfert plus lente sur les réseaux mobiles – Possiblité de déconnection lors de la navigation – Autonomie de l'équipement (batterie)
Puissance de calcul	– Processeurs moins performants – Mémoire vive moins importante
Applicatif	– Support partiel des standards W3C par le navigateur – Absence de lecteur Flash

TAB. I.1 – *Catégories de limitations des appareils mobiles et des STB*

Cependant les limitations des plates-formes ne sont pas les seules à appeler une adaptation des documents Web. La diversité des utilisateurs et de l'environnement les entourant est aussi une raison pour l'adaptation.

Or le verbe adapter est transitif: on adapte un élément à une "cible" donnée. Il convient alors de définir ce qu'est une cible, puisque la cible est intrinsèquement la source des contraintes à apposer pour l'élément à adapter. Cela nous permettra de différencier les contraintes qui appellent à l'adaptation. Nous l'avons emprunté à (Thevenin, 2001) :

Définition. Une **cible** se définit par le triplet <classe d'utilisateurs, plate-forme, environnement> où:

- "plate-forme" désigne le support matériel et logiciel qui sous-tend l'interaction. La taille de l'écran, les dispositifs d'interaction, les capacités de calcul et de communication caractérisent une plate-forme d'interaction tels un PC, un PDA ou un téléphone portable.
- "environnement" dénote le milieu dans lequel s'exerce l'interaction homme-machine. Il comprend un ensemble d'informations, périphériques à la tâche en cours, mais susceptibles de l'influencer [...] : le lieu géographique ou symbolique (à domicile, au cinéma, dans la rue, dans le train), le jour (ouvrable, férié), l'heure (de jour, de nuit), l'ambiance (fréquenté, bruyant).

Le but des hypermédias adaptatifs est donc d'avoir une interface multi-cible donc capable de s'adapter à toute cible.

Par sa nature potentiellement réflexive et selon les interprétations, il est possible de concevoir l'adaptation comme la possibilité pour un document Web de s'adapter à la cible, caractérisant les hypermédias adaptatifs ou comme l'action d'adapter un document Web déjà existant à une cible. Il convient donc de faire la distinction entre ces deux approches.

Définition. L'**adaptation *a priori*** ou **adaptativité** est la capacité pour un document Web de s'adapter à une ou plusieurs cibles.

Définition. L'**adaptation *a posteriori*** est l'action d'un système qui adapte un document Web pour une cible différente de celle pour laquelle il a été conçu.

Nous avons fait le choix de ces définitions car la dichotomie dynamique/statique ne nous paraissait pas reflêter la réalité des travaux de Thevenin (2001) où l'interface est générée de façon dynamique à partir d'une abstraction.

I.1.2 Les positions de l'adaptation

Si le besoin d'adapter les documents Web est réel, il reste à déterminer à quel niveau de l'architecture client-serveur l'adaptation va avoir lieu. À quelle étape de la livraison du contenu informatif, entre le serveur hébergeant l'information et le client l'affichant, positionner le processus d'adaptation des documents Web?
Le positionnement a un impact fort. Par exemple, si l'adaptation se fait sur le client, il

est nécessaire d'enrichir celui-ci. Au contraire, s'il se fait entre le client et le serveur, par un tiers, celui-ci devra avoir accès à l'ensemble du contenu Web et aux informations sur la cible.
Afin de catégoriser les positions d'adaptation, on peut faire ressortir trois tendances principales :

- La première est celle d'assurer l'adaptation au niveau du serveur Web donc du fournisseur de documents Web.
- A l'opposé, il est possible que l'adaptation ait lieu sur le dispositif du côté client.
- Au niveau intermédiaire, l'adaptation peut être prise en charge par un serveur mandataire.

Nous nous proposons donc de détailler les différentes places où peut avoir lieu l'adaptation. Il est à noter que l'adaptation peut être distribuée entre deux places voire les trois. Plus d'informations à propos du partage de l'adaptation sur plusieurs places sont données dans la partie I.2.1.2.

Adaptation côté serveur

Le premier positionnement possible dans la chaîne de livraison du contenu est l'adaptation côté serveur. En effet, avant même de servir le contenu, le serveur peut l'adapter. L'adaptation des documents au niveau de leurs fournisseurs existe déjà pour de nombreux sites commerciaux [2]. Le principal inconvénient est que ces sites ont souvent, en pratique, seulement deux versions, l'une classique pour un ordinateur de bureau et une version mobile pour un terminal à la mode par exemple. Pourtant cette vision dichotomique a l'avantage de proposer des versions créées en connaissant pleinement la cible, et donc présentant souvent une restructuration des informations pour suivre au mieux les contraintes des plates-formes. Au rang des défauts figure le fait que le développement et l'entretien de plusieurs versions ont un coût élevé. De plus l'existence d'une version mobile est tributaire de la volonté du fournisseur de la développer, n'offrant aucune garantie aux utilisateurs sur l'existence d'une telle version.

Adaptation côté client

Parmi les limitations des dispositifs clients, nous avons mis en avant les processeurs moins puissants et la mémoire vive limitée. A partir de ce constat, il est clair que l'adaptation au niveau du client est pénalisée par le manque de puissance de calcul. La capacité de la batterie est aussi un facteur limitant à prendre en compte pour l'adaptation. Cependant l'avantage de l'adaptation sur le client est la possibilité de relever aisément les caractéristiques de l'environnement. En effet, en étant au plus proche de l'utilisateur, le client est le mieux placé pour connaître les capacités d'affichage du terminal, les interfaces de

2. On peut citer "Gmail sur votre portable", "Youtube XL" ou "Yahoo! Mobile".

commande à disposition et relever des données de capteurs. En effet, des capteurs intégrés ou rattachés au dispositif peuvent apporter des éléments supplémentaires sur l'environnement par exemple. Les préférences de l'utilisateur peuvent facilement être demandées et stockées en limitant les problématiques liées à la vie privée. Ainsi les caractéristiques du dispositif, les préférences de l'utilisateur et les données supplémentaires venant des capteurs sont des bases solides pour l'adaptation. Pourtant, la multiplicité des environnements logiciels et donc des dévelopements nécessaires à un déploiement de masse de solutions d'adaptation positionnées sur le client sont à ajouter aux inconvénients cités auparavant.

Adaptation au niveau du serveur mandataire

Entre la source et la destination du contenu, il est possible d'opérer l'adaptation. Pour cela, le flux devra transiter par un entité logicielle appelée serveur mandataire. Le serveur mandataire est aussi appelé communément par l'anglicisme proxy. Cela dénote seulement la distinction de ce logiciel avec le logiciel serveur et le logiciel client. En effet, ce serveur mandataire peut être hébergé par une machine distincte mais peut également être hébergé sur la même machine que le client ou le serveur. A noter que, s'il est implanté du côté client, il sera soumis aux mêmes limitations physiques. Cependant, cette solution s'avère beaucoup plus souple, puisqu'elle permet l'hébergement du processus d'adaptation n'importe où dans la chaîne de livraison. Cette possible indépendance vis-à-vis des dispositifs de navigation implique que le serveur mandataire se doit de prendre en considération la multiplicité des dispositifs. Par conséquent, le principal problème à résoudre est celui de connaître les caractéristiques des dispositifs lors de l'adaptation. De plus, si le serveur mandataire est transparent, il présente l'inconvénient d'adapter les documents Web que les utilisateurs le veuillent ou non. Enfin le fait d'imiscer un tiers dans les transactions entre le client et le serveur présente un réel danger pour la vie privée de l'utilisateur, le proxy étant spectateur de toute information non-cryptée.

Comparaison des positions d'adaptation

La table I.2 résume les avantages et inconvénients de chacune des places où peut se tenir l'adaptation.

Il apparaît dans le tableau I.2 que certaines faiblesses d'une place d'adaptation peuvent être compensées par les avantages d'un ou des deux autres placements. Ceci explique d'autant plus la distribution de l'adaptation sur plusieurs composantes de la chaîne de livraison des documents Web. Il est possible de cumuler les avantages de deux ou des trois places d'adaptation, à la condition de pouvoir maîtriser la partie impliquée (client ou serveur) et leur coordination.

Lieu de l'adaptation	Avantages	Inconvénients
Serveur	– Existe déjà pour nombre de sites commerciaux – Qualité de l'adaptation pour les versions développées spécifiquement pour une cible connue par avance	– Coût de développement et d'entretien – Le plus souvent, deux versions : les solutions ne sont pas universelles mais spécifiques à deux cibles seulement – Au bon vouloir du fournisseur de documents
Client	– Accès direct aux caractéristiques du dispositif – Stockage des préférences utilisateur aisé – Possibilité d'accès direct à des capteurs pour déterminer l'environnement d'utilisation	– Limitée par la puissance de calcul – Tributaire de l'autonomie du terminal – Plusieurs versions de code pour chaque famille de dispositif
Serveur mandataire	– Indépendant des contraintes et limites du serveur et du client – Peut adapter le document selon plusieurs plates-formes	– Nécessité de connaître les caractéristiques du dispositif-cible – En mode transparent, absence de contrôle sur l'adaptation – Vie privée potentiellement menacée

TAB. I.2 – *Forces et faiblesses des places d'adaptation.*

I.1.3 Bilan

Cette première partie a permis de dépeindre une vue d'ensemble de la problématique de la consultation de pages Web sur les équipements hétérogènes à l'heure de l'informatique pervasive (PDA, téléphone portable, tablet, TV via SetTop Box, ...). En effet ces équipements, même s'ils offrent la possibilité de consulter les informations du Web sans les contraintes d'un ordinateur, sont intrinsèquement limités. Afin de rendre la navigation sur la Toile la plus agréable possible, il est souhaitable que la page soit adaptée aux limitations des équipements mais aussi aux préférences de l'utilisateur et le cas échéant, aux contraintes de l'environnement lors de la consultation.

Le terme adaptation peut avoir deux sens. Le premier est la capacité d'un document Web à prendre plusieurs formes, selon les contraintes. Le second sens est l'action d'adapter un

contenu à ces contraintes. Dans la suite de ce manuscrit, nous poursuivons par l'étude de l'existant, tant au niveau scientifique que technique, sur l'adaptation dynamique par un système tiers des documents Web.

I.2 Etat de l'art

L'objectif de cette partie est l'étude bibliographique sur l'adaptation dynamique de documents Web. Celle-ci s'appuie sur les définitions et les classifications présentées dans la partie précédente.

La section I.2.1 consiste en une étude des travaux académiques sur l'adaptation dynamique.

Dans la section I.2.2, nous présentons les systèmes d'adaptation.

I.2.1 Travaux académiques

I.2.1.1 Périmètre de l'étude

Cette étude bibliographique des travaux académiques a porté sur la dernière décennie. En effet il nous paraît plus pertinent de mettre en évidence les travaux récents pour montrer l'originialité de nos travaux plutôt que de remonter plus loin et de complexifier davantage la lecture de cette étude.

En ce qui concerne le domaine, nous avons limité notre champ de recherche aux travaux d'adaptation des documents Web. De plus nous nous sommes attachés à ne citer que la plus récente des publications pour chaque projet.

La comparaison complète des 47 publications est disponible grâce aux tables I.3 et I.4.

Afin de présenter de manière synthétique notre étude, celle-ci s'articule autour de deux critères principaux :

- la position de l'adaptation : du côté serveur, client et/ou grâce à un serveur mandataire.
- la méthode d'adaptation : transcodage, ré-ingénierie ou navigation côté client

Pour des raisons de concision, l'analyse sur les pays du ou des laboratoires étant à l'origine des publications ainsi que l'analyse sur les années de parution des publications ont été reportées en annexe A.

	Année	**Pays**	**Prise en compte du contexte**	**Place de l'adaptation**	serveur	client	proxy	**Méthode d'adaptation**	Transcodage	*textuel*	*image*	*audio et video*	*Flash*	Re-engineering	*Reverse-engineering*	*Forward-engineering*	Navigation côté client
Adipat & Zhang (2005)	2005	Etats-Unis				✓	✓			✓							✓
Arase *et al.* (2007)	2007	Japon				✓	✓			✓	✓						✓
Ardon *et al.* (2003)	2003	Australie	✓				✓				✓	✓					
Artail & Raydan (2005)	2005	Liban	✓		✓					✓	✓						
Baluja (2006)	2006	Etats-Unis					✓			✓	✓						
Berhe *et al.* (2005)	2005	France	✓				✓			✓	✓						
Berti & Paternò (2005)	2005	Italie	✓		✓											✓	
Bickmore *et al.* (1999)	1999	Etats-Unis					✓			✓	✓						
Björk *et al.* (1999)	1999	Suède				✓	✓			✓							✓
Borodin *et al.* (2007)	2007	Etats-Unis					✓			✓							
Bouillon *et al.* (2005)	2005	Belgique					✓								✓	✓	
Burigat *et al.* (2008)	2008	Italie				✓											✓
Buyukkokten *et al.* (2002)	2002	Etats-Unis					✓			✓							
Casteleyn *et al.* (2006)	2006	Belgique			✓											✓	
Cena *et al.* (2006)	2006	Italie	✓		✓											✓	
de Oliveira & da Rocha (2007)	2007	Brésil				✓				✓							✓
De Virgilio *et al.* (2007)	2007	Italie/Belgique	✓				✓			✓	✓						
Di Lucca *et al.* (2005)	2005	Italie													✓		
Ferretti *et al.* (2007)	2007	Italie					✓			✓							
Gaeremynck *et al.* (2003)	2003	Etats-Unis													✓		
Grassel *et al.* (2006)	2006	Finlande				✓				✓							✓
Gupta *et al.* (2006)	2006	Etats-Unis					✓			✓	✓						✓
jung Han *et al.* (2009)	2009	Corée du Sud									✓						

TAB. I.3 – *Comparaison des publications (première partie)*

	Année	**Pays**	**Prise en compte du contexte**	**Place de l'adaptation**	serveur	client	proxy	**Méthode d'adaptation**	Transcodage	*textuel*	*image*	*audio et video*	*Flash*	Re-engineering	*Reverse-engineering*	*Forward-engineering*	Navigation côté client
Hori *et al.* (2000)	2000	Japon/Etats-Unis			✓		✓			✓							
Hübsch *et al.* (2005)	2005	Allemagne			✓											✓	
Hwang *et al.* (2003)	2003	Corée du Sud					✓			✓	✓						
Jones *et al.* (1999)	1999	Royaume-Uni				✓				✓							✓
Joshi (2000)	2000	Etats-Unis	✓		✓	✓	✓				✓	✓					
Komninos & Milligan (2007)	2007	Royaume-Uni				✓				✓							
Korva *et al.* (2001)	2001	Finlande	✓			✓	✓									✓	
Lam & Baudisch (2005)	2005	Canada/Etats-Unis				✓	✓			✓							✓
Lunn *et al.* (2008)	2008	Royaume-Uni				✓				✓							
MacKay *et al.* (2004)	2004	Canada				✓											✓
Maheshwari *et al.* (2002)	2002	Inde	✓				✓			✓	✓						
Mahmud *et al.* (2009)	2009	Etats-Unis													✓	✓	
Milic-Frayling *et al.* (2004)	2004	Royaume-Uni				✓	✓			✓							✓
Mohomed *et al.* (2006)	2006	Canada	✓			✓	✓			✓	✓						
Moshchuk *et al.* (2008)	2008	Etats-Unis					✓						✓				
Otterbacher *et al.* (2008)	2008	Chypre/Etats-Unis					✓			✓							
Paganelli & Paternò (2003)	2003	Italie													✓		
Song & Lee (2008)	2008	Corée du Sud			✓											✓	
Watters & Zhang (2003)	2003	Canada			✓					✓							
West (2005)	2005	Etats-Unis					✓			✓							✓
Wobbrock *et al.* (2002)	2002	Etats-Unis				✓											✓
Xiao *et al.* (2009)	2009	Chine	✓		✓					✓							
Xie *et al.* (2005)	2005	Chine				✓	✓										✓
Yin & Lee (2005)	2005	Singapour								✓							

Tab. I.4 – *Comparaison des publications (dernière partie)*

I.2.1.2 Analyse

Diagrammes de répartition

Par position de l'adaptation Le premier axe pour analyser les publications retenues est la position de l'adaptation. En effet, dans le paradigme client-serveur, l'adaptation peut intervenir au niveau du client, du serveur ou d'un serveur mandataire, aussi nommé par l'anglicisme proxy. De plus, l'adaptation peut être partagée par deux positions voire les trois. La figure I.1 présente les positions d'adaptation des travaux présentés par les articles sélectionnés.

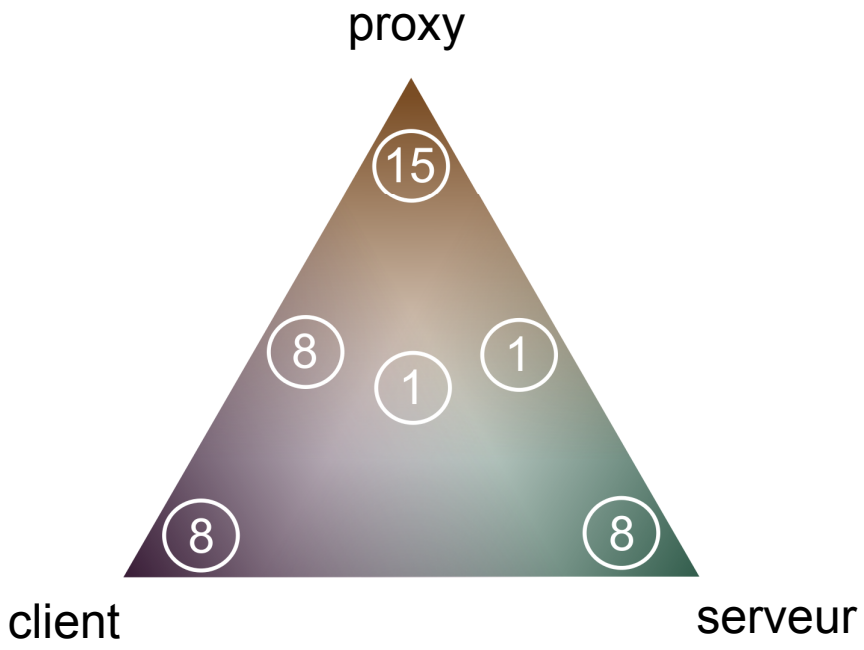

FIG. I.1 – *Répartition de notre étude en fonction du lieu d'adaptation : nombre d'articles par position d'adaptation.*

Dans chaque angle du triangle figurent les nombres de projets faisant partie de cet état de l'art et dont l'adaptation s'appuie seulement sur l'hôte correspondant. Au contraire, les chiffres à mi-chemin entre deux sommets du triangle figurent le nombre de travaux qui ont opté pour une adaptation partagée entre les deux composants-sommets. Enfin, le numéro au milieu du triangle représente le nombre de travaux partageant l'adaptation sur les trois composants.

Les publications présentent, pour environ un tiers (15/47 pour être précis), une adaptation hébergée par un serveur mandataire. Un autre tiers (16 articles sur les 47) est partagée à part égale entre les serveurs et les clients comme hôtes de l'adaptation. Les approches d'adaptation partagées entre deux composantes, quant à elles, ne représentent que 9 articles dont les auteurs de 8 d'entre elles ont optés pour une adaptation partagée entre le client et le proxy. De plus, notons que l'approche partagée entre le client et le serveur n'est pas représentée dans cette étude. Enfin, un seul travail scientifique (Joshi, 2000) s'est penché sur l'adaptation distribuée entre les trois composantes.

Pour résumer, l'étude de la position de l'adaptation pour notre état de l'art montre un plébiscite de l'utilisation de serveurs mandataires ainsi que des approches partagées qui l'utilisent, comptabilisant 25 articles, c'est-à-dire plus de la moitié des publications sélectionnées.

Par méthode d'adaptation Avant de voir cette classification, il convient de définir ce que nous entendons par méthode d'adaptation. D'après (Gerber *et al.*, 2002), la transformation d'une interface graphique (notée IG) pour un autre contexte peut se faire par deux chemins différents. La figure I.2 présente ces deux méthodes. Sur la partie inférieure de la figure, le transcodage qui est une transformation directe de code à code, s'opposant à la ré-ingénierie, faite de trois étapes principales :

- la rétro-ingénierie ou ingénierie inverse qui consiste à asbtraire l'interface graphique pour obtenir un modèle sous-jacent,
- la transformation modèle à modèle qui passe d'un modèle abstrait d'interface à celui pour un autre contexte
- l'ingénierie (en anglais forward-engineering) qui, à partir du nouveau modèle, construit une interface graphique pour le nouveau contexte.

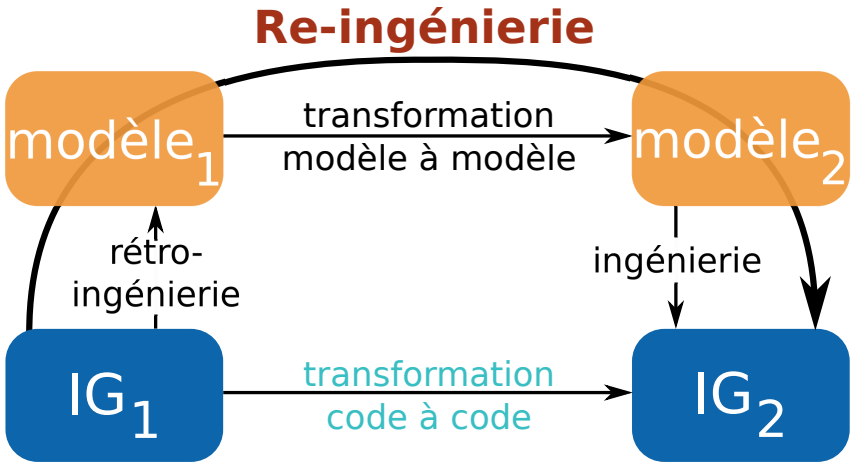

FIG. I.2 – *2 méthodes d'adaptation : le transcodage et la ré-ingénierie.*

En plus de ces deux méthodes d'adaptation (Lardon *et al.*, 2008), nous avons aussi retenu la navigation côté client comme une troisième méthode. Dans ce cas, l'adaptation est différente au sens que la transformation ne se fait pas sur l'interface graphique mais sur la façon de la présenter. Comme exemple de navigation côté client, nous pouvons citer les zooms, systèmes de vignettes (thumbnails en anglais) ou de loupes.

A l'image des positions d'adaptation, nous avons séparé les cumuls des publications faisant seulement appel à une méthode d'adaptation de ceux faisant appel à la composition de deux méthodes. La figure I.3 présente la répartition des publications retenues en fonction de leur méthode d'adaptation.

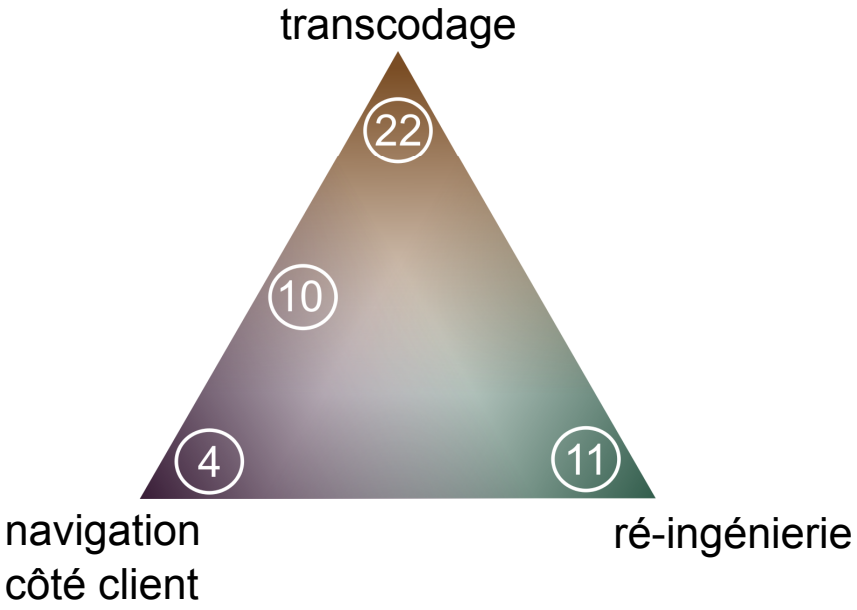

FIG. I.3 – *Répartition de notre étude selon la méthode d'adaptation: nombre d'articles par méthode d'adaptation.*

Il ressort de notre étude que le transcodage est la méthode la plus utilisée par les travaux listés dans notre état de l'art avec 22 travaux sur 47. Viennent ensuite les 11 travaux faisant appel à la ré-ingénieurie pour adapter les contenus Web. Enfin, 4 articles utilisent seulement la navigation côté client. On remarquera aussi qu'une seule composition de ces méthodes apparaît parmi les publications sélectionnées, à savoir la composition du transcodage et de la navigation côté client qui représente 10 travaux scientifiques.

I.2.2 Systèmes d'adaptation

Nous présentons dans cette partie un panel des projets existant pour l'adaptation de contenu Web. Nous avons pris le parti de distinguer les projets en deux catégories : les projets académiques et les projets industriels.

I.2.2.1 Projets académiques

Comme nous l'avons vu dans la section précédente, l'adaptation de contenu Web a été le domaine de nombreuses recherches scientifiques à travers le monde. Certaines d'entre elles ont de plus débouché sur des prototypes de systèmes d'adaptation. Nous présentons par la suite ceux que nous avons jugés les plus pertinents.

TranSend

Ce proxy de l'université de Berkeley (Fox *et al.*, 1999) a pour but d'adapter le contenu au changement de réseaux. En effet, lors du passage d'un réseau à un autre, la bande passante du nouveau réseau peut diminuer. En conséquence, le contenu délivré aux équipements sur ce réseau devrait être réduit pour conserver la fluidité de navigation. C'est dans ce but que TranSend intègre un module de réception de paquets UDP dédiés à la notification des nouvelles caractéristiques du réseau. Ces notifications sont envoyées par un logiciel client installé sur l'équipement à chaque changement de réseau. Le proxy est donc en charge d'adapter le contenu selon les caractéristiques du réseau.

L'inconvénient majeur de TranSend vient du fait que l'adaptation est limitée aux images, que ce soit la compression et le redimensionnement des images ou la réduction de l'espace de couleur. De plus, ce projet ne prend pas en compte les préférences des utilisateurs.

Digestor

Digestor (Bickmore *et al.*, 1999) vise à restructurer les documents du Web pour qu'ils soient affichés sur des équipements mobiles à petit écran. Le concept de "re-authoring" y est introduit. Nous traduirions ce terme par ré-écriture des contenus Web. Cette ré-écriture, proche d'une re-structuration, se veut automatique et est effectuée par un proxy. De plus, les techniques de ré-écriture automatique sont catégorisées selon deux dimensions :

- syntaxique ou
- sémantique

et

- transformation ou
- élision

A l'aide de plusieurs techniques de re-structuration, Digestor se propose de sélectionner la meilleure des compositions de techniques pour que le document résultant tienne au mieux dans l'espace limité d'affichage.

La critique que l'on pourrait apporter à ce système est de plusieurs ordres :

1. Les techniques d'adaptation proposées sont toutes de type syntaxique.
2. La sélection de la meilleure composition passe par l'utilisation d'heuristiques dont le potentiel est limité (Buchholz *et al.*, 2003).
3. L'adaptation n'est guidée que par la taille de l'écran et non pas par toutes les caractéristiques de l'équipement ou l'état du réseau.

4. L'utilisateur doit spécifier la taille de l'écran de son équipement et la taille de police de son navigateur par défaut.

Mowser

Mowser (contraction de mobile et browser) (Joshi, 2000) est un serveur mandataire chargé de transcoder les documents dynamiquement. Au contraire des deux projets précédents, Mowser permet aux utilisateurs de saisir leur préférences ainsi que les caractéristiques de leur équipement. Ces informations ne servent pas exclusivement pour l'adaptation mais aussi pour modifier les requêtes HTTP. En effet, le serveur mandataire ajoute à la requête un paramètre pour demander au serveur Web une variante du document. A condition que le serveur supporte ce système de variantes, il retourne une variante qui devrait correspondre aux informations fournies par l'utilisateur. Dans le cas où le document retourné par le serveur prend une taille à l'écran supérieure à celle disponible sur le terminal, une adaptation a effectivement lieu au niveau du proxy.

Le désavantage majeur de Mowser est le fait que les informations utilisateur soient liées à une adresse IP. Le système de variantes présente aussi deux inconvénients :

- une empreinte mémoire conséquente, du fait de la multiplication des versions et donc des documents présents sur le serveur
- l'absence de garantie que ce système soit supporté par le serveur Web et le client.

Power Browser

Power Browser (Buyukkokten *et al.*, 2002) est un système de transcodage hébergé par un proxy. Ce système n'a, par contre, que pour cible des équipements affichant ligne par ligne. L'adaptation passe exclusivement par le résumé. Les textes ainsi que les formulaires sont résumés pour être affichés sur un Palm Pilot. La page Web est découpée en "Semantic Textual Units" (STUs) dont seulement les premiers mots sont affichés, certaines STUs pouvant en contenir d'autres. L'utilisateur peut, par la suite, afficher progressivement le reste du texte d'une STU, directement tout le texte de la STU ou prendre connaissance des STUs contenues. Ce système de développement et réduction ("expand and collapse" en anglais) des STUs est baptisé accordéon[3]. Quant aux formulaires, le résumé de ceux-ci se fait en n'affichant que les intitulés des champs. Les auteurs ont donc dû développer des algorithmes pour faire correspondre chaque label avec le ou les champs correspondants.

Les inconvénients de ce système sont au nombre de deux :

1. Le manque de généricité par rapport aux équipements.

3. Terme retrouvé dans les nouvelles fonctionnalités de la version 3 des Cascading Style Sheets

2. L'adaptation est seulement au niveau textuel : les images sont remplacées systématiquement par leur attribut HTML ALT.

DCAF

DCAF (Berhe *et al.*, 2005) est un système distribué d'adaptation de contenus Web. Du point de vue de l'utilisateur, ce système semble être hébergé par un proxy mais il est bien plus complexe. En effet, derrière ce proxy local, au plus proche du client mais jamais sur le poste client, l'architecture de DCAF regroupe plusieurs éléments :

- des services d'adaptation répertoriés dans un registre,
- des proxys de contenu en amont des serveurs de contenu et
- un gestionnaire de profils utilisateurs.

Les proxys locaux sont les points centraux de cette architecture car ils sont en charge de requêter le gestionnaire de profils utilisateurs et le registre des services d'adaptation pour choisir la meilleure composition d'adaptation.

Le principal défaut de ce système tient à sa dépendance aux services d'adaptation et aux proxys de contenu. En effet, les proxys de contenu sont chargés, en plus de la liaison entre les proxys locaux et les serveurs de contenu, de fournir des métadonnées sur les contenus.

Crunch

Crunch (Gupta *et al.*, 2006) est présenté comme une boîte à outils pour l'extraction automatique de contenu. En cela, il peut paraître éloigné de notre problématique. Pourtant ce système agit exclusivement sur des pages Web et utilise des filtres en guise de transformations. Ainsi les éléments ajoutés au contenu principal de la page Web (publicités, menu, etc...) peuvent être replacés à la fin de la page ou bien la page entière peut être transformée pour n'avoir que du texte en sortie. Par la suite, ce texte pourra éventuellement être dicté par un logiciel. Le principal intérêt de Crunch est qu'il est possible de déployer des modules qui sont chargés du filtrage de la page. Par conséquent, chacun peut développer et intégrer ses propres filtres.

Au rang des désavantages de Crunch, on peut compter :

1. la prise en compte exclusive des données d'une page Web, aucune transformation sur les images ou autres ressources,
2. la notion de filtre (transformation possiblement avec perte),
3. l'absence de personnalisation : tous les utilisateurs d'une même instance de Crunch sont voués à avoir les mêmes options pour Crunch,

4. aucune recherche de la meilleure composition de filtres : les filtres sont mis dans un ordre par les options et donc l'ordre est le même quelle que soit la page.

SADIe

SADIe (Lunn *et al.*, 2008) est un système de transcodage côté client à destination des mal-voyants et des non-voyants. Le transcodage des pages repose sur l'annotation manuelle des feuilles de style CSS pour déterminer l'importance de chacun des éléments de la page. Une fois que l'annotation est faite, elle est réutilisable, puisque les feuilles de style sont très souvent partagées par plusieurs pages Web d'un même site.

Quatre méthodes de transcodage sont utilisées dans SADIe :

1. **Defluff** consiste à enlever les éléments ayant peu ou pas de relation avec le contenu principal de la page.
2. **Reorder** réordonne les informations de la plus importante en haut de la page, à la moins impotante en bas de la page.
3. **Menu** replace le menu de la page à la fin de celle-ci pour permettre d'accéder plus rapidement au contenu principal
4. Enfin, **Concertina** affiche un aperçu de certains éléments et offre la possibilité de développer et de réduire ces aperçus pour accéder à la totalité de ces éléments

Pourtant, SADIe présente deux désavantages. Le premier est l'aspect manuel de l'annotation. Le second inconvénient de ce système vient du fait qu'il s'exécute du côté client. Donc son utilisation sur n'importe quel terminal est difficilement envisageable dû aux limitations computationnelles éventuelles du terminal.

I.2.2.2 Projets industriels

Par la multiplication des équipements de consultation du Web, le domaine de l'adaptation automatique de contenu n'est pas seulement l'apanage de la communauté scientifique. En effet, nombre de projets industriels d'adaptation ont vu le jour. Cette partie a pour but d'en lister les principaux ayant retenu notre attention.

Web Clipping

Ce produit de Palm Computing a vu le jour en 1998 et vise à adapter les pages HTML en vue d'un affichage sur le PDA Palm VII. Ce proxy intervient autant pour ajuster la taille de la page que pour limiter la bande passante utilisée. L'adaptation passe par deux étapes : la suppression des éléments redondants et la transformation de la page pour correspondre à la taille de l'écran. La consommation de ressources réseau est contrôlée grâce à la transmission pas à pas des éléments de la page. Ainsi, les éléments sélectionnés dès le début pour être affichés sont transmis directement. Ceux apparaissant par la suite ne

seront transmis qu'à la demande de l'équipement.

Il apparaît clairement que ce système souffre de son manque de généricité, car il a été développé pour l'affichage sur un seul modèle de PDA.

IBM Transcoding Proxy

Ce proxy de transcodage, développé par le "Mobile Networking Group" d'IBM, a pour but l'adaptation multimédia pour l'affichage sur des équipements de petite taille, comme les PDA et les téléphones mobiles. Deux modes d'adaptation sont supportés par ce système. Un premier pour les PDA consiste à résumer les textes, à transformer les images en leur équivalent noir et blanc et à analyser les éléments audio et video pour les remplacer par un équivalent textuel. Le second mode d'adaptation a pour cible les téléphones mobiles et donc remplace les textes par un titre, supprime les images et donne une version audio des contenus vidéos.

Le principal défaut de ce système est son fonctionnement figé. En effet la configuration est manuelle et ne peut donc pas prendre en compte les facteurs dynamiques comme la bande passante du réseau. De plus, les options de configuration sont limitées dans le sens où les transformations sur les images ne comprennent pas un changement de format.

WebSphere Transcoding Publisher

Autre développement d'IBM, WebSphere Transcoding Publisher est aussi un proxy de transcodage mais sous forme d'un frontal aux serveurs Web. Par rapport à son prédécesseur, ce système prend en compte les préférences utilisateurs et les caractéristiques des équipements. Nombre de méthodes de transcodage ont été ajoutées, telles que la transformation de tables en listes ou la substitution des images par des liens pour les afficher en dehors de la page. De plus, le transcodage porte sur d'autres formats que les formats basiques HTML, CSS, GIF, par exemple VoiceXML ou WBMP. Cette généricité est d'autant plus avancée que le système offre la possibilité de développer et de brancher des modules de transcodage.

Pourtant, ce système de module est restrictif dans le sens où le développement de ces modules se fait grâce à un kit de développement propriétaire, ce qui limite fortement l'interopérabilité avec des développements existants.

Mobile Firefox

La version mobile de Firefox[4] ne propose pas, comme les projets précédents, de transcodage, mais seulement des outils de présentation en guise d'adaptation. En premier lieu,

4. https://wiki.mozilla.org/Mobile

ce navigateur affiche un aperçu du document Web (page HTML ou image) prenant la largeur de l'écran. Par la suite, il est possible de zoomer ou dézoomer pour voir plus précisément une partie du document. De plus les barres d'adresse, d'onglet et de navigation ne gènent pas la navigation puisqu'elles sont placées respectivement en haut, à gauche et à droite du document. Ainsi, lorsque l'utilisateur fait glisser le document vers le bas et qu'il atteint le haut de celui-ci, apparaît la barre d'adresse. Il en est de même pour les barres d'onglet ou de navigation qui ne sont affichées que lorsque l'utilisateur va plus loin que la gauche ou la droite du document.

Un inconvénient majeur cependant : lors de la rédaction de ce manuscrit, les équipements supportés sont au nombre de deux, à savoir les terminaux Nokia N900 et Nokia N810 ainsi qu'une version d'émulation pour ordinateurs de bureau.

Nokia Open Source Browser

Ce second navigateur est l'aboutissement de Grassel *et al.* (2006). Il propose une adaptation de la largeur des textes pour que ceux-ci ne prennent pas plus que la largeur de l'écran. Ainsi l'utilisateur n'est-il pas obligé de faire défiler horizontalement la page pour lire un texte. De plus, plusieurs fonctions de navigation sont présentes. Lors du défilement horizontal ou vertical, un aperçu de la page avec un cadre représentant la zone affichée apparaît en sur-impression dans l'angle en haut à droite. En outre, la navigation entre les pages déjà explorées pendant la session se fait par des aperçus miniatures entre lesquels l'utilisateur navigue avant de sélectionner et revoir une page précise.

Comme le projet précédent, ce navigateur est pénalisé par le nombre limité d'équipements supportés.

Opera Mini

L'adaptation du navigateur Opera Mini[5] n'intervient pas au niveau du client. Des proxy hébergés par Opera Software sont effectivement en charge d'adapter le contenu requêté pour l'afficher au mieux sur l'équipement. Cette adaptation revient à fractionner la page en plusieurs sous-pages et à mettre le contenu de la page en forme pour tenir une seule colonne de la largeur de l'écran de l'équipement. Un tel découpage permet une navigation fluide en permettant un affichage rapide des pages grâce aux proxy chargés également du rendu des pages et qui les transmettent au client dans le langage OBML.

Opera Mini présente pourtant les limites suivantes :

1. Les utilisateurs sont dépendants des serveurs mandataires d'Opera Software.

5. http://www.opera.com/mini/

2. Les contenus dynamiques (animations GIF ou flash) sont au mieux converties en images fixes, au pire non pris en charge.
3. Le javascript est supporté mais pour chaque action, la page doit être rechargée.

I.2.3 Bilan

Nous dressons ici le bilan de notre étude bibliographique. Dans un premier temps, nous nous sommes attachés à faire un tour d'horizon des travaux académiques relatifs à l'adaptation dynamique de documents Web sur les dix dernières années. En sont ressorties 47 publications scientifiques. Sur ces publications retenues, nous avons analysé leur répartition selon deux critères : (1) par position d'adaptation (client, serveur ou proxy) et (2) par méthode d'adaptation (transcodage, ré-ingénierie ou navigation côté client).

Par la suite, nous avons étudié une sélection des projets existants, qu'ils soient l'aboutissement d'une recherche académique ou une solution industrielle, mettant en exergue leurs limites.

Tout d'abord les publications suivent majoritairement l'adaptation par transcodage. Cette méthode d'adaptation présente à nos yeux plusieurs limitations. La première est le nombre limité de techniques de transcodage de certaines publications. Vient ensuite le problème de la composition des techniques : Est-elle possible ? Est-elle paramétrable ? Si elle est automatique, comment l'ordre est-il décidé ? C'est à ces questions que les travaux ayant un nombre conséquent de techniques ont du mal à répondre efficacement. Parmi ceux qui n'utilisent pas le transcodage, nous ne retrouvons quasiment que des travaux qui exécutent l'adaptation au niveau du serveur ou du client. En cela, ils nous paraissent limités. Si le client est chargé de faire l'adaptation, il n'est pas garanti que l'équipement aura assez de capacités (puissance de calcul, mémoire, etc.). De même, l'adaptation du côté serveur limite les ressources Web accessibles via l'équipement de l'utilisateur à celles servies par le fournisseur.

De plus, les solutions d'adaptation dynamique des documents Web frappent par leur manque de généricité. Ceci est d'autant plus vrai pour les offres commerciales de navigateur donc fortement dépendantes du client. Même dans le cas des systèmes reposant sur un serveur mandataire, la configuration peut être fastidieuse ou alors toutes les caractéristiques des équipements de navigation ne sont pas pris en compte. Cela étant dit, le serveur mandataire nous paraît l'approche la plus astucieuse puisqu'il est potentiellement plus proche du client que le serveur (typiquement sur le même segment réseau) pour une meilleur prise en compte du contexte dans les adaptations, sans subir les contraintes d'une adaptation côté client (puissance de calcul, autonomie, etc.). Cependant, il est aussi possible de placer le serveur mandataire comme frontal à un ou plusieurs serveurs, per-

mettant par là même de répartir la charge de l'adaptation.

Enfin l'extensibilité n'est pas le point focal des systèmes d'adaptation. Si certains n'offrent pas la possibilité d'ajouter de nouvelles techniques d'adaptation, d'autres permettent d'ajouter des greffons mais présentent des tâches d'ingénierie associées très lourdes ou limitent la portée d'action des greffons.
Etant donné ces observations sur les travaux déjà effectuées dans le domaine de l'adaptation dynamique des documents Web, nous avons fait les choix suivants pour notre propre travail :

- Nous utiliserons un proxy comme hôte de l'adaptation.
- Le transcodage sera notre méthode d'adaptation.

Tel que nous l'envisageons, un système d'adaptation dynamique de documents Web idéal satisferait les contraintes suivantes :

1. indépendance par rapport à l'équipement ciblé et aux documents à adapter
2. temps d'exécution de l'adaptation imperceptible pour l'utilisateur
3. utilisabilité optimale pour le résultat de l'adaptation

L'originalité et l'apport de nos travaux par rapport à l'état de l'art est double :

- universalité par rapport au dispositif (le dispositif faisant une requête au proxy est considéré comme non déterminé par avance : la cible d'adaptation n'est pas connue d'une quelconque heuristique par avance)
- étude d'une nouvelle approche dans la conduite de l'adaptation (cf. chapitre 2)

Dans le cadre de notre thèse, nous nous sommes fixés les objectifs suivants.

Objectif 1 : Indépendance par rapport aux cibles

- Le système ne doit rien connaître des cibles par avance.
- Le système doit récupérer les caractéristiques de la cible par conséquent.

Objectif 2 : Non-intervention humaine

Le système doit pouvoir adapter n'importe quelle page Web en fonction des caractéristiques de la cible de manière automatique.

Objectif 3 : Adaptation en un temps raisonnable

Le système doit effectuer l'adaptation d'une page en un temps le plus court possible, de manière à ne pas perturber la navigation de l'utilisateur.

Objectif 4 : Réalisation technologique

Le système doit être implémenté afin de valider empiriquement le modèle proposé.

CHAPITRE I

Problématique et état de l'art

Conclusion

Ce premier chapitre a eu pour but de poser la problématique de l'adaptation automatique des documents Web et de présenter un tour d'horizon des publications scientifiques et des projets dans ce domaine.

La première partie de ce chapitre a traité la problématique de l'adaptation. En plus de soulever les motivations pour adapter les documents Web, nous nous sommes attachés à recenser les mécanimes contemporains d'adaptation.

A partir de cette étude, les publications dans le domaine de l'adaptation automatique de documents Web ont été catégorisées. De plus, une sélection de systèmes d'adaptation a été discutée afin de souligner les limites actuelles.

Enfin, à partir des faiblesses des approches existantes, nous avons formulé les objectifs de nos travaux qui sont la généricité, l'universalité et un temps d'exécution raisonnable de notre mécanisme d'adaptation.

CHAPITRE II

Modélisation d'une nouvelle approche d'adaptation

Sommaire

CHAPITRE II

Modélisation d'une nouvelle approche d'adaptation

Introduction

Dans le chapitre précédent, nous avons mis en évidence les freins au développement des systèmes d'hypermédia adaptatifs existants. C'est-à-dire le dilemme de l'emplacement dans le paradigme client-serveur du mécanisme d'adaptation, le manque de mise en œuvre réelle et donc manque d'évaluation, ainsi que l'absence de généricité des modèles mentionnés. C'est ainsi que nous avons notamment dégagé les objectifs d'universalité et de généricité pour notre proposition de modèle d'hypermédia adaptatifs.

Il s'agit d'adresser les situations d'informatique pervasive de plus en plus nombreuses puisque le rythme effréné des sorties des nouveaux dispositifs de navigation ne permet plus de prévoir par avance des versions de document Web, adaptées ou transposables, spécifiques à ces nombreux équipements hétérogènes en terme de plate-forme et d'interface homme-machine.

Ce chapitre 2 présente le modèle que nous proposons pour adresser la pervasivité des hypermédia adaptatifs. Nous exposons tout d'abord une vue synoptique de notre modèle en terme d'architecture et déclinons les éléments qui nous paraissent importants à définir pour saisir le fonctionnement de celui-ci. Ensuite, nous motivons l'emploi de méta-heuristiques en démontrant que notre modèle pose un problème d'optimisation étant donné la taille de l'espace de recherche dans l'automatisation du processus d'adaptation. Nous présentons successivement l'architecture et le fonctionnement algorithmique des trois grandes composantes de notre modèle.

II.1 Modélisation

Dans cette partie, nous décrivons l'approche générale proposée puis nous présentons succintement le modèle du processus d'adaptation que nous avons développé.

II.1.1 Approche générale

A partir des objectifs posés dans le bilan du chapitre 1 (I.2.3), la première tâche à laquelle nous nous sommes attelés fut le choix du positionnement de notre système dans le paradigme client-serveur.

La solution d'un serveur mandataire fut choisi à plusieurs égards :

1. le dispositif client peut ne pas avoir la capacité de mener à bien dans un temps court l'adaptation (puissance, batterie, plate-forme logicielle, etc.),
2. l'hébergement par le serveur de l'adaptation limiterait le nombre de documents adaptés à ceux hébergés par le serveur,
3. le serveur mandataire a l'avantage de pouvoir être positionné n'importe où dans la chaîne de livraison des documents Web : comme passerelle des clients, comme relais inverse en frontal d'un ou plusieurs serveurs, ou comme service choisi par l'utilisateur.

Pourtant le serveur mandataire présente un désavantage comme hôte de l'adaptation à l'instar de l'adaptation côté serveur : le contexte n'est pas directement accessible comme il pourrait l'être en plaçant l'adaptation du côté client. La réponse à ce problème est abordée dans la partie II.2.

La question de la localisation de l'adaptation ayant été arbitrée, il reste à définir comment l'adaptation sera faite. Nous avons sélectionné le transcodage. Par rapport aux autres travaux effectués à l'aide de cette méthode d'adaptation, nous nous distinguons par le fait que nous utilisons des estimations dans le processus d'adaptation. De plus le choix et l'ordre des techniques d'adaptation mises en œuvre sont optimisés par une méta-heuristique.

Les estimations citées plus loin ont pour vocation de servir de support pour les techniques de transcodage ainsi que de guider la méta-heuristique dans le choix de la meilleure suite de techniques. Ces estimations interviennent sur l'arbre DOM[1] d'un document Web, liant à chaque nœud de l'arbre une ou plusieurs estimations.

Avant d'expliciter notre modèle d'adaptation, nous revenons sur les notions d'arbre DOM, de propriété visuelle et de propriété caractéristique, d'estimation et d'estimateur, de transformation, et de contexte.

1. Document Object Model

Arbre DOM (notion d'ordre générale)

Un arbre DOM[2] est une représentation arborescente d'un document HTML et plus généralement XML. L'arbre DOM est construit en menant une analyse grammaticale sur un document HTML ou XML. Pour cela, un nœud est créé pour chaque élément ou texte du document. Les éléments ou textes contenus par un élément donneront lieu à des nœuds qui seront les fils du nœud correspondant à l'élément contenant.

Propriété visuelle, propriété caractéristique (notions propres à notre modèle)

Avant de pouvoir définir une estimation, il est nécessaire de définir le concept de propriété visuelle et de propriété caractéristique.

> *Définition.* Une **propriété visuelle** d'un nœud est une propriété dont la valeur influence l'affichage de ce nœud par un navigateur.

Nous pouvons donner comme exemple de propriétés visuelles la liste suivante :

- la mise en couleur d'un passage de texte dans un document Web,
- le formatage horizontal du nœud : marges extérieures et intérieures et bordures horizontales, ainsi que largeur de l'élément ou du texte correspondant au nœud.

> *Définition.* Une **propriété caractéristique** d'un nœud est une propriété dont la valeur n'influence pas l'affichage du nœud mais permet d'enrichir les informations sur ce nœud.

Parmi les propriétés caractéristiques, nous pouvons citer :

- la profondeur du nœud dans l'arbre DOM,
- le nombre de mots contenus dans le sous-arbre dont le nœud est la racine, pour limiter la vocalisation à des textes assez longs,
- un booléen distinguant si le nœud fait partie d'un lien HTML (balise <A> avec un attribut HREF),
- le nombre de mots contenus dans le sous-arbre dont le nœud est la racine mais aussi à l'intérieur d'un lien HTML, pour éviter de vocaliser un texte contenant des liens.

Estimation et estimateur (notions propres à notre modèle)

> *Définition.* Une **estimation** est une valeur approchée d'une propriété visuelle ou d'une propriété caractéristique pour un nœud.

Les quatre propriétés caractéristiques données comme exemples suivent des types primitifs (entier, booléen, flottant). Au contraire, le second exemple de propriété visuelle regroupe un ensemble de paramètres et doit être stockée dans une structure plus complexe. C'est

2. http://www.w3.org/TR/DOM-Level-2-Core/introduction.html

pour cela que les estimations sont possiblement très complexes en structure, bien qu'elles n'approchent que les valeurs de chaque paramètre.

> *Définition.* Un **estimateur** est une composante logicielle visant à calculer une estimation pour chaque nœud d'un arbre DOM.

Une explication plus complète sur les estimateurs et le processus qui permet de calculer les estimations est disponible dans la partie II.3 de ce manuscrit.

Transformation (notion propre à notre modèle)

> *Définition.* Une **transformation** est une implémentation d'une technique de transcodage, ayant deux fonctions : simuler leur action sur les estimations, et transformer effectivement l'arbre DOM. Les conditions d'application d'une transformation sur un nœud donné sont dépendantes des estimations ou d'une liste de nœuds-cibles.

Contexte

Le contexte est défini, dans notre modèle, comme le groupement des deux premiers éléments constituant une cible dont la définition empruntée à (Thevenin, 2001) est rappelée dans la partie I.1.1. Nous définissons donc le contexte comme le couple <classe d'utilisateurs, plate-forme>, abstraction étant donc faite de l'environnement. Cette restriction est la conséquence de notre objectif de généricité. En effet, une minorité de terminaux de navigation embarque des capteurs capables de capturer les informations liées à l'environnement.

> *Définition.* Le **contexte** est un couple <classe d'utilisateurs, plate-forme> où :
>
> – "classe d'utilisateurs" dénote les informations à propos de l'utilisateur,
>
> – "plate-forme" est la définition matérielle et logicielle sous-tendant l'interaction

II.1.2 Modélisation du processus d'adaptation

Les concepts d'estimateurs, de transformations et de contexte ayant été expliqués, il est possible de représenter le processus d'adaptation que nous proposons à l'aide de la figure II.1.

Le processus d'adaptation prend en entrée le document Web et le contexte, mais aussi une liste d'estimateurs et une liste de transformations. Le processus peut donc être résumé en 5 grandes étapes :

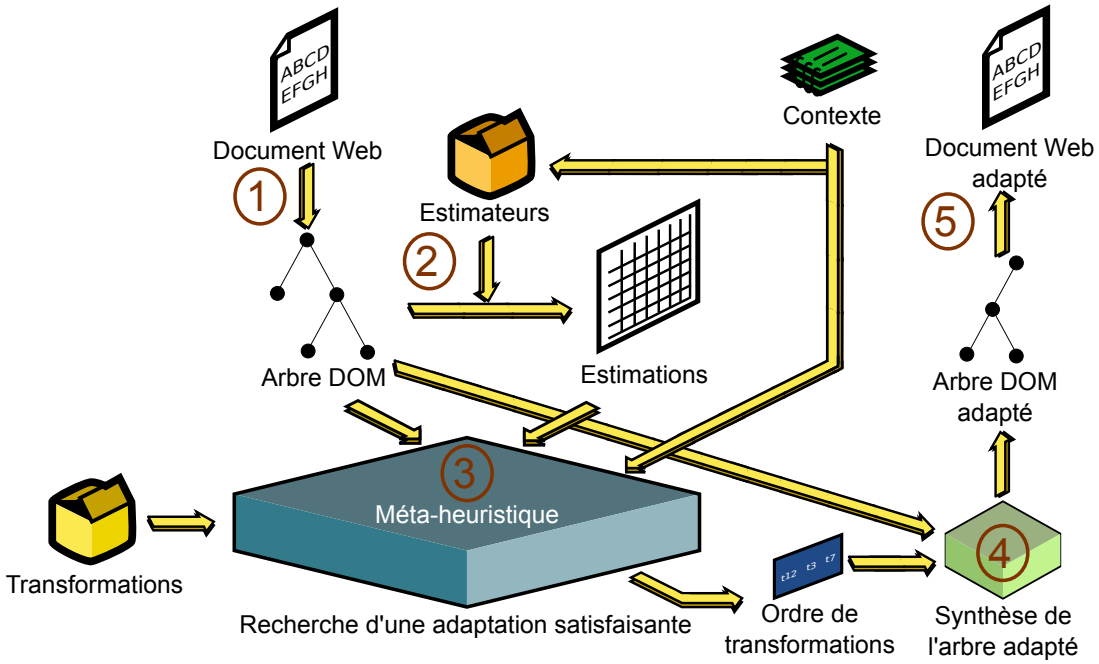

FIG. II.1 – *Modèle du processus d'adaptation.*

1. création de l'arbre DOM à partir du document Web,
2. estimation des valeurs sur tous les nœuds de l'arbre DOM à l'aide des estimateurs,
3. recherche d'une suite de transformations satisfaisante grâce à une méta-heuristique,
4. synthèse de l'arbre DOM adapté à partir de l'arbre DOM original et de l'ordre des transformations,
5. création du document Web adapté suivant l'arbre DOM adapté.

Plutôt que de regrouper l'ensemble des étapes dans un processus unique, nous avons pris le parti de le distribuer en trois modules. L'architecture proposée est présentée dans la figure II.2 et est composé de trois modules (proxy de contexte, module de session et moteur de transformation) brossés rapidement par la suite.

Les estimateurs et les transformations mises à part, notre système nécessite comme entrées le document Web et le contexte. Le bloc chargé de récupérer le document Web ainsi que le contexte est appelé **proxy de contexte** et fait l'objet de la partie II.2.

Par la suite, la construction de l'arbre DOM, l'enregistrement des estimateurs et des transformations, ainsi que la première estimation, sont faits au niveau du **module de session**, couvert par la partie II.3.

Enfin, la méta-heuristique au cœur du **moteur de transformation**, est étudié plus en détail dans la partie II.4.

Le processus, découpé selon les trois modules, est présenté dans la figure II.3.

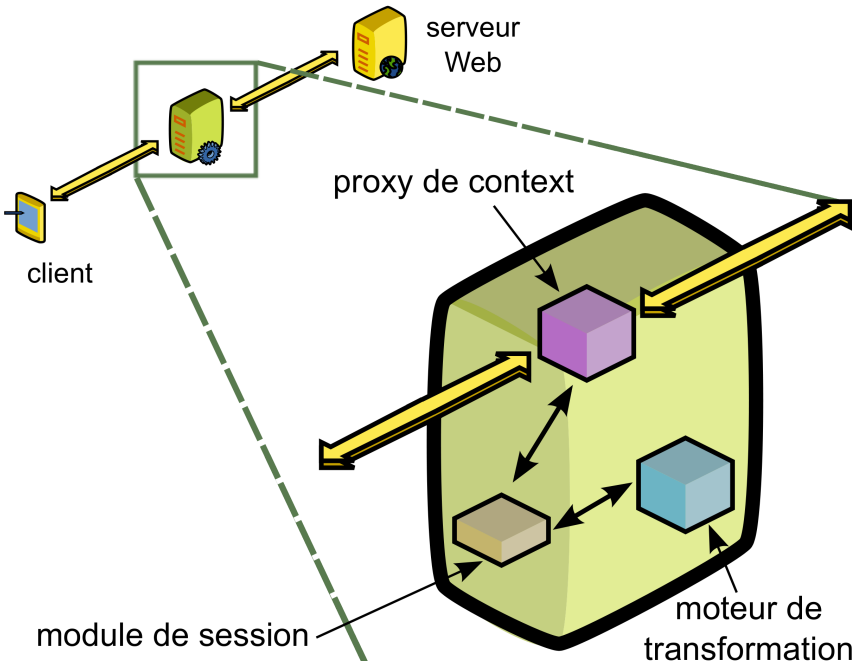

FIG. II.2 – *Schéma bloc de notre système.*

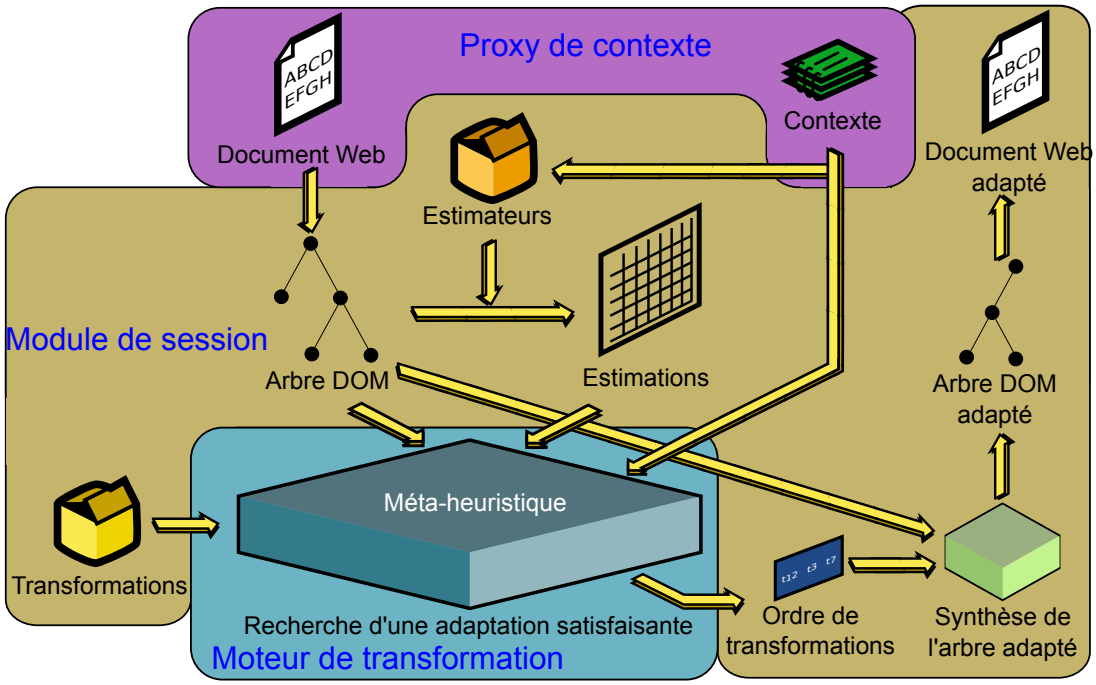

FIG. II.3 – *Blocs de notre système au niveau du processus d'adaptation.*

II.2 Proxy de contexte

Le proxy de contexte gère les échanges entre les clients et les serveurs Web en charge la récupération et la propagation du contexte.

L'aspect communication de ce bloc comprend la réception des requêtes des utilisateurs et leur retransmission aux serveurs hébergeant les documents Web. De plus, la réception des réponses des serveurs ainsi que l'emission des réponses à destination des utilisateurs lui sont dévolues.

En plus de cela, le proxy de contexte est chargé de récupérer et de propager le contexte d'utilisation. En effet, le problème pour pouvoir adapter au mieux un document Web est de savoir dans quel contexte d'utilisation prend place la navigation. Ainsi, lorsque le système reçoit une requête, il est nécessaire de savoir dans quel contexte elle a été émise. Des solutions existent déjà pour lier un contexte d'utilisation à une session de navigation Web. Cependant, nous avons développé notre propre mécanisme de récupération et de propagation du contexte. Ce mécanisme s'appuie sur la technologie des cookies HTTP [3].

Afin d'expliquer les raisons qui nous ont poussés à mettre en œuvre ce mécanisme, il est nécessaire de faire un rapide état des solutions apportées par les travaux académiques qui ont précédé.

II.2.1 Étude des moyens de liaison entre le contexte et la session de navigation Web

Des solutions ont été proposées pour établir un lien entre la session de navigation et le contexte. Elles peuvent être classifiées en quatre catégories :

1. Authentification HTTP : à chaque session, l'utilisateur s'authentifie.
2. Identification par adresse Internet : lors de la connexion, l'adresse IP sert de référence pour connaître l'équipement et donc une partie ou le contexte complet
3. Identification par entête HTTP : en lisant les entêtes HTTP, il est possible d'avoir une idée de l'équipement dont est issue la requête
4. Navigateur riche : si le client est capable, par exemple, d'ajouter des entêtes HTTP pour inclure des éléments du contexte, le système d'adaptation sera capable de lire directement ces informations.

L'authentification HTTP est un des moyens mis en œuvre pour connaître le contexte lors d'une session de navigation. Elle est par exemple utilisée dans DCAF (Berhe *et al.*, 2005) comme première étape de la navigation. L'authentification offre l'avantage de faire remonter un profil utilisateur, d'équipement et/ou d'environnement de façon détermi-

3. http://www.ietf.org/rfc/rfc2965.txt

niste. Cependant, si la saisie d'un couple identifiant/mot de passe est une chose aisée et quotidienne sur un ordinateur de bureau, cela peut devenir fastidieux à l'aide d'une télécommande pour téléviseur et sans l'aide d'un système de gestion de mot de passe par exemple. Un mot de passe permet à l'usager d'indiquer un contexte. Il faut donc autant de mots de passe par utilisateur qu'il peut exister de contexte. Si le contexte se limite au type de périphérique et que, préalablement, il est associé sur le serveur, pour un utilisateur, un mot de passe par équipement, il est nécessaire que l'utilisateur retienne autant de mots de passe qu'il a d'équipements.

Une seconde solution pour lier le contexte à une requête est l'utilisation de l'adresse IP comme référence, à l'image de Mowser (Joshi, 2000). Cette solution a l'avantage de la simplicité puisque l'adresse IP de l'équipement émettant la requête est disponible lors de la connexion avant que les entêtes HTTP ne soient lues. Le principal défaut de cette solution est son manque de stabilité. En effet même si l'adresse IP est unique, un équipement connecté en DHCP [4] peut avoir une adresse IP différente chaque fois qu'il est connecté au réseau. En outre, l'utilisation d'un serveur mandataire, qu'il ait pour but de filtrer ou d'anonymiser, empêche de différencier les équipements qui y font appel, puisqu'ils partagent tous la même adresse IP sur Internet. Enfin, l'adresse IP pourrait permettre au mieux de connaître l'équipement émetteur de la requête, mais en aucun cas le profil utilisateur ni les informations concernant l'environnement de navigation.

Le système de migration de Berti & Paternò (2005) fait appel aux entêtes HTTP pour différencier les types d'équipements, via l'entête *UserAgent*. Pourtant cette entête n'a pas été conçue pour véhiculer plus d'informations que celles sur les navigateurs et éventuellement quelques informations sur le système d'exploitation. Le fait est que ces informations peuvent, au plus, donner des indications sur l'équipement. Assurément, cette solution ne donne pas assez d'informations (par exemple la taille de l'écran) pour permettre une granularité plus fine d'adaptation.

Enfin, la dernière solution consiste à recourir à des navigateurs riches. On peut citer les navigateurs capables de récupérer les informations sur l'équipement et le transmettre sous la forme de profils CC/PP [5], à l'instar des navigateurs utilisés pour TranSquid (Maheshwari *et al.*, 2002) et MARCH(Ardon *et al.*, 2003). Quand bien même cette solution apporte une partie de la solution, l'objectif de généricité que nous nous sommes imposés ne peut être mené à bien en ignorant les navigateurs classiques, plus passifs que les navigateurs riches.

4. Dynamic Host Configuration Protocol
5. Composite Capability/Preference Profiles

II.2.2 Notre approche : les cookies HTTP

Les cookies HTTP apparaîssent comme un moyen intéressant pour stocker les informations de contexte. Le mécanisme de cookie a fait ses preuves dans des systèmes d'adaptation tels que Skweezer[6]. De plus, les cookies ont l'avantage d'être stockés sur le poste client et envoyés pour chaque requête sur un domaine donné. Par conséquent, les informations de contexte sont fortement liées à l'équipement. Pourtant, cet avantage peut être une limitation pour maintenir plusieurs profils utilisateur au niveau d'un même équipement.

> *Définition.* Le **cookie de contexte** est un cookie HTTP contenant les informations liées à l'équipement et au profil utilisateur. Il existe sous différentes formes dont celle d'un identifiant unique pointant sur les informations de contexte contenues dans une base de données.

La figure II.4 illustre l'adaptation d'un document Web dont l'adresse **A** fait partie du domaine ***D***. Il est supposé, dans ce cas, que le cookie de contexte existe sur l'équipement client pour le domaine ***D***. Comme nous le verrons dans la partie suivante, les cookies sont liés à un domaine et ne sont adjoints à la requête que si celle-ci porte sur une adresse du domaine.

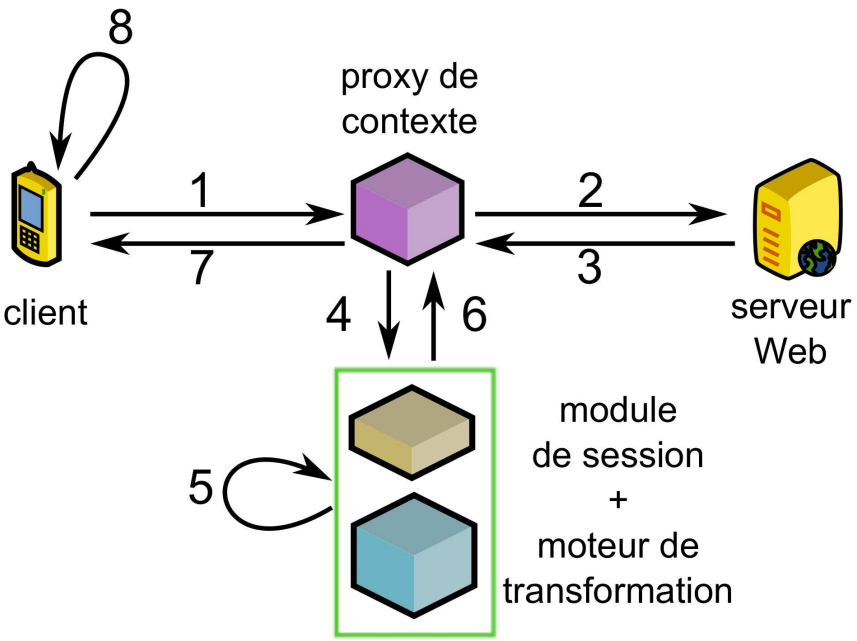

FIG. II.4 – *Adaptation d'un document Web si le cookie de contexte existe pour son domaine.*

Les étapes de la figure II.4 sont les suivantes :

1. Le client envoie au système une requête pour le document d'adresse **A** sur le domaine ***D***. Cette requête contient le cookie de contexte.

6. http://www.skweezer.com/

2. Le proxy de contexte reçoit la requête, en fait une copie, supprime le cookie de contexte de la copie, et finalement, fait suivre la copie au serveur hébergeant le document Web souhaité.
3. Le serveur retourne le document au proxy de contexte.
4. Le proxy de contexte fournit au module de session le cookie de contexte ainsi que le document Web.
5. A l'aide des informations contenues dans le cookie de contexte, le document Web est adapté.
6. Le document adapté est retourné au proxy de contexte.
7. Le proxy de contexte fait suivre le document adapté au client. La réponse HTTP contenant le document adapté hérite des entêtes de la réponse de l'étape 3, à l'exception des entêtes *Set-cookie* pouvant changer tout ou partie du cookie de contexte.
8. Le navigateur client affiche le document adapté.

Cependant, ce processus est possible à la condition de l'existence du cookie de contexte pour le domaine ***D***. Dans le cas contraire, il est nécessaire de propager le cookie de contexte de domaine en domaine pour couvrir ceux sur lesquels l'utilisateur navigue.

II.2.3 Cookie de contexte trans-domaine

Comme l'adaptation a été présenté dans la figure II.4, un cookie universel – pour tous les domaines – aurait été une solution parfaite. Néanmois, les cookies sont limités à un domaine ou un sous-domaine par définition. C'est la raison pour laquelle le cookie de contexte doit être propagé d'un domaine à l'autre ou plus exactement d'un domaine de référence aux autres.

Pour mettre en place cette propagation, nous nous sommes inspirés du mécanisme de *cross-domain cookie*. Dans notre modèle, un domaine sert de référence pour le cookie de contexte. Ce domaine est nommé "domaine maître" comme traduction de l'anglicisme *master domain* et est noté ***MD***. De plus, il est nécessaire d'avoir un serveur sur ce domaine, appelé serveur maître et noté **MS**.

En reprenant le cas précédent, mais en supposant que le cookie de contexte existe sur le domaine ***MD*** et non sur le domaine ***D***, l'adaptation a le même début : la requête de l'étape 1 ne contient pas de cookie de contexte, les étapes 2 et 3 sont identiques au cas précédent. L'étape 4 devient :

4. Le proxy de contexte met la réponse du serveur en cache, puis émet une redirection HTTP vers le serveur maître **MS** sur le domaine maître ***MD***. L'adresse de redirection contient un paramètre α identifiant de manière unique l'adresse de la requête originale (**A**). L'adresse de redirection est donc de la forme `http://serveur.maitre/?alpha=(.*)`.

Par la suite, deux boucles de redirection sont utilisées pour propager le cookie de contexte vers le domaine ***D***. La figure II.5 permet de décomposer ces deux boucles.

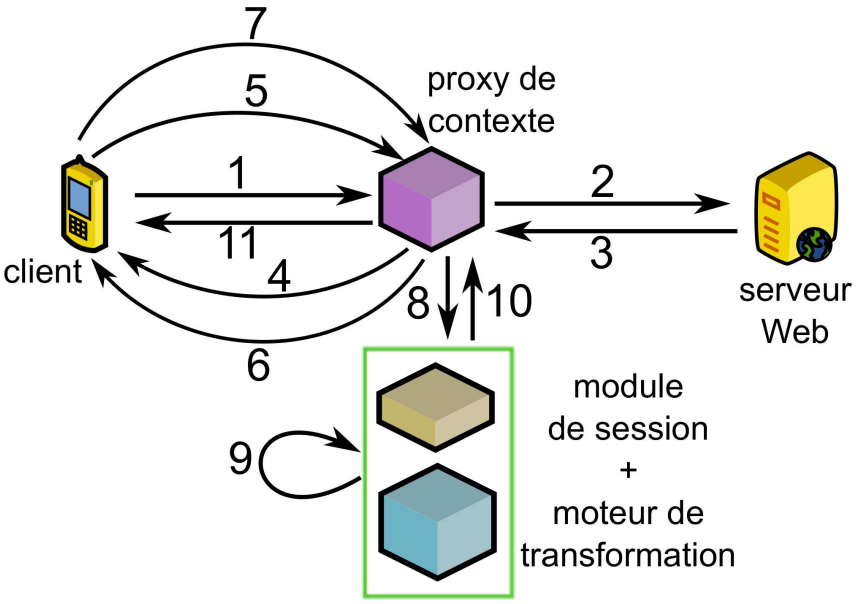

FIG. II.5 – *Processus de propagation du cookie de contexte du domaine maître à un autre domaine.*

5. Le navigateur reçoit l'instruction de redirection et émet une requête pour le serveur maître avec, dans les entêtes, le cookie de contexte.
6. Le proxy de contexte reçoit la requête pour le serveur maître et répond par une redirection vers l'adresse **A**. L'adresse **A** est retrouvée grâce au paramètre α de l'étape 4. L'adresse de redirection contient le cookie de contexte comme paramètre de requête GET.
7. Le navigateur reçoit la seconde redirection vers l'adresse initiale. Le navigateur produit en conséquence la requête GET de l'adresse **A** avec le contexte en paramètre.
8. Le proxy de contexte reçoit la requête vers le serveur d'origine et le cookie de contexte. Il récupère dans le cache le document enregistré à l'étape 4 et l'envoie avec les informations de contexte au module de session.
9. A l'aide des informations contenues dans le cookie de contexte, le document Web est adapté.
10. Le document adapté est retourné au proxy de contexte.

II.2.4 Récupération du contexte et création du cookie sur le domaine maître

Même lors de la propagation, il est supposé que le cookie de contexte existe pour le domaine maître. Or ce ne peut être le cas lors de la toute première requête, après la suppression du cookie de contexte ou suite à son expiration. Il convient donc de récupérer les informations du contexte afin de construire le cookie de contexte.

Ceci est réalisé grâce à une redirection. Cette boucle s'intercalle entre l'étape 5 et l'étape 6 du processus de propagation. Comme la figure II.6 le montre, cette boucle se divise en 2 étapes :

5 bis Le proxy de contexte retourne un document Web capable de détecter et/ou proposant de saisir les données de contexte, par exemple les caractéristiques de l'équipement et les préférences de l'utilisateur.

5 ter Du côté client, le navigateur et/ou l'utilisateur recueille(nt) les informations du contexte et une requête à une adresse dédiée appartenant au domaine maître ***MD*** est envoyée avec les informations de contexte comme paramètres d'adresse0.

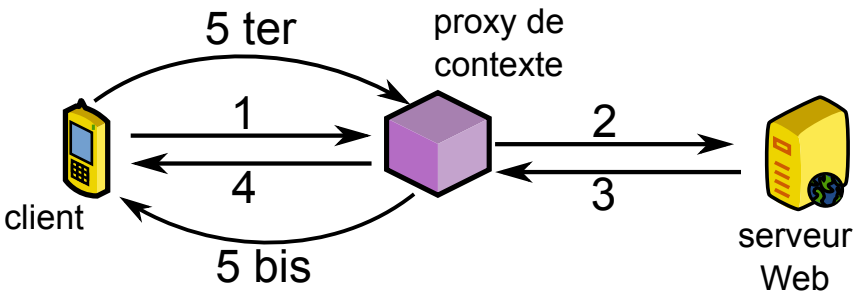

FIG. II.6 – *Processus de création du cookie de contexte pour le domaine maître.*

Dans ce cas, les informations de contexte ne sont pas tirés du cookie de contexte mais des paramètres GET de la requête envoyée à l'étape 5ter. De plus, une entête *Set-cookie* est ajoutée à la redirection de l'étape 6 afin d'assigner le cookie de contexte pour le domaine maître.

Lors de l'étape 5 ter, la récupération du contexte est effectué grâce à un script rédigé en langage Ecmascript[7] ou renseigné par l'utilisateur grâce un formulaire (Lardon *et al.*, 2010b). Le détail de l'implémentation est présenté dans la partie III.1.

II.2.5 Bilan

Le proxy de contexte a pour but de gérer les communications entre le client et les serveurs hébergeant les documents Web. De plus, il a la charge de garder disponible le contexte

7. http://www.ecmascript.org/docs/tc39-2009-043.pdf

d'utilisation tout au long de la session de navigation. A cet effet, nous avons choisi d'utiliser des cookies HTTP pour stocker le contexte lors de la navigation.

L'utilisation de cookies HTTP offre l'avantage d'être générique puisque faisant partie des standards du Web émis par l'IETF[8]. De plus, le stockage dans un cookie HTTP est effectué du côté client ce qui place l'information au plus près de l'utilisateur. Le fait que les cookies soient liés à un domaine aurait pu être un inconvénient mais avec le système de propagation présenté, cette limitation est levée et permet de couvrir au fur et à mesure les domaines sur lesquels navigue l'utilisateur.

Au rang des inconvénients, on peut citer la possibilité sur de nombreux navigateurs de désactiver les cookies. La redondance de l'information est aussi un handicap qui peut s'avérer problématique si le cookie de contexte contient l'ensemble des données du contexte et si l'équipement bénéficie de peu de mémoire. Cela tend cependant à disparaître avec les nouvelles versions de smartphones et STB embarquant une grande capacité mémoire. L'empreinte mémoire de ces systèmes revêt donc une importance plus réduite qu'auparavant. De plus, en l'état, ce système ne supporte pas plusieurs profils utilisateurs sur un même équipement. Cependant, on peut tout de même relever une limitation dûe au fait que les boucles de redirection servant à propager le cookie de contexte à partir du domaine maître peuvent s'avérer coûteuses en temps dans un réseau dont la latence est forte.

Cette section a fait état de l'architecture ayant mené à l'implémentation du proxy de contexte durant cette thèse.

Une amélioration est envisageable. Il est possible d'avoir plusieurs profils utilisateurs liés à un équipement, à condition que les profils soient stockés indépendamment des autres données du contexte. Il serait possible de lier plusieurs profils à un même identifiant de contexte contenu dans le cookie. Pour cela, il faudrait, en début de session, proposer à l'utilisateur les différents profils liés au contexte.

La récupération du document Web et du contexte ainsi que la persistance du contexte lors de la navigation étant expliqués, il reste à aborder l'adaptation en elle-même. Le processus d'adaptation est partagée entre deux blocs : le module de session et le moteur de transformation.

8. Internet Engineering Task Force

II.3 Module de session

Afin de rester le plus générique possible, il est nécessaire d'intercaler un composant entre le proxy de contexte et le moteur de transformation. Ce bloc a été appelé module de session. En effet, le système d'adaptation a été développé comme ouvert, avec la possibilité de développer et d'ajouter des transformations. Un obstacle a été identifié lors de la conception : comment assurer la continuité de la navigation lorsqu'une transformation découpe un document Web en plusieurs sous-pages ? Le premier objectif du module de session est d'assurer cette continuité. De plus, ce bloc factorise certains processus pour l'adaptation des sous-pages potentielles. Enfin, il est aussi chargé de réaliser la synthèse du document Web adapté à l'aide de l'ordre de transformations retourné par le moteur de transformation.

Le nom de ce bloc est déduit de ses fonctions. Pourtant, si le document est découpé, la navigation entre les sous-pages est assimilée à une session de navigation à part entière.

Cette partie est organisée comme suit. Tout d'abord, nous décrivons la mise en place d'une session avec les processus communs à toutes les sous-pages. Par la suite, nous expliquons le mécanisme de continuité dans la navigation entre sous-pages.

II.3.1 Création d'une session

Une session est définie par le couple <document Web, contexte>. C'est lors de la création de la session que les estimateurs et les transformations sont enregistrés. Ensuite a lieu l'analyse grammaticale du code du document pour construire un arbre DOM. Il est important d'enrichir l'arbre DOM avec le style de chaque nœud. C'est ainsi qu'en plus d'avoir accès au nœud, il est indispensable d'avoir rapidement accès à son style, qu'il soit pris de l'attribut HTML *style* ou qu'il vienne du style du document. Une fois cela fait, l'estimation initiale des valeurs de l'arbre DOM a lieu. Avant d'expliquer comment se fait l'estimation, il est nécessaire de définir plus exactement ce qu'est un estimateur.

Estimateurs

Un estimateur est un composant logiciel capable de calculer la valeur approchée d'une propriété visuelle ou d'une propriété caractéristique pour chacun des nœuds d'un arbre DOM. Pour cela, il s'appuie sur le contexte, ainsi que sur les nœuds qui sont en relation avec le nœud concerné, c'est-à-dire son nœud-père et/ou ses fils. Pour illustrer les trois catégories d'estimateurs, trois exemples sont proposés.

Le premier exemple est celui de la profondeur du nœud. Cette estimation est simple car sa valeur est exacte. En effet, la profondeur de la racine de l'arbre est 0 et les autres nœuds ont une pronfondeur égale à la profondeur de leur nœud-père incrémenté de 1.

Le second exemple est l'estimation du nombre de mots contenus par le nœud et par ses fils éventuels. Au contraire de l'exemple précédent, cette estimation se fait par rapport aux fils du nœud concerné, voire lui-même si le nœud est un texte. En effet l'estimation pour un nœuds ayant des fils est la somme des estimations des fils.

Enfin le dernier exemple est, à notre sens, le plus parlant, le plus complexe mais aussi le plus important pour l'adaptation. Cette estimation est celle des propriétés horizontales d'un nœud. Ces propriétés regroupent les marges extérieures et intérieures, les largeurs de bordures et la largeur du nœud. Ces propriétés sont par exemple représentées par les proriétés CSS[9] *margin*, *padding*, *border* et *width*. Pour calculer l'estimation des propriétés horizontales d'un nœud, il faut avoir recours à son père et/ou à ses fils selon les nœuds. Pour estimer le plus justement ces propriétés, nous avons utilisé les spécifications[10, 11] du W3C[12] pour les langages HTML[13] 4.0 et CSS 2.
Par exemple, pour un élement de type TABLE, chaque cellule, ligne et tableau a une taille dépendante aussi bien de son nœud-père que de ses nœuds-fils.

Les trois catégories d'estimateurs sont appelés estimateurs préfixes, suffixes et mixtes respectivement pour les 3 estimateurs précédents : (1) la pronfondeur du nœud, (2) le nombre de mots dans le sous-arbre dont le nœud est la racine et (3) les propriétés horizontales. Les estimateurs préfixes nécessitent donc le concours de l'estimation du nœud-père. Les estimateurs suffixes font appel aux estimations des fils pour calculer celle du père. Enfin, les estimateurs mixtes adoptent, selon le nœud, un comportement préfixe ou suffixe.

En outre, un estimateur peut être dépendant d'un ou de plusieurs autres estimateurs. C'est le cas de l'estimateur de propriétés horizontales. En effet, sans connaître les valeurs des propriétés CSS *display* et *float* pour le nœud, il est impossible de savoir si l'estimation des propriétés horizontales doit être calculée par rapport au nœud père ou aux nœuds fils. La nécessité d'un estimateur pour un autre est indépendante des catégories d'estimateurs présentés auparavant. En effet, quelle que soit la catégorie d'une estimateur, il peut être lié à un ou plusieurs autres estimateurs.

Ainsi, lors de l'enregistrement des estimateurs, le système vérifie au fur et à mesure si les dépendences des estimateurs sont vérifiées. De même, les transformations sont dépendantes d'estimateurs pour leurs conditions d'application ou pour la simulation de leur action. C'est pour cela que le système vérifie lors de leur enregistrement que chacun des estimateurs dont dépendent les transformations sont bien enregistrés.

Une fois tous les estimateurs chargés en mémoire, le module de session effectue une

9. Cascading Style Sheet
10. http://www.w3.org/TR/1998/REC-html40-19980424/
11. http://www.w3.org/TR/CSS2/
12. Word Wide Web Consortium
13. HyperText Markup Language

première estimation complète de l'arbre DOM.

Processus d'estimation

Les estimations d'un nœud étant fonction des autres estimations de ce nœud, de celles de son père ou de ses fils, le processus d'estimation suit le principe d'un parcours en profondeur traitant les estimateurs préfixes avant de passer à l'estimation des enfants puis les estimateurs postfixes après l'estimation des enfants.

Ce parcours commence au nœud BODY pour aller au bout de chacune des branches. L'algorithme 1 est celui utilisé pour assurer le processus d'estimation.

Algorithme 1 Algorithme d'estimation des valeurs d'un nœud : la fonction *estimer*

ENTRÉES: n un nœud, pre la liste des estimateurs préfixes, suf la liste des estimateurs suffixes, mix la liste des estimateurs mixtes

```
{On effectue les estimations préfixes}
estimation(pre, n)
{On effectue les estimations mixtes ayant un comportement préfixe pour le nœud n}
estimation_pre(mix, n)
{On estime les valeurs pour les nœuds fils}
pour tout f ∈ fils(n) faire
  estimer(f, pre, suf, mix)
fin pour
{On effectue les estimations suffixes}
estimation(suf, n)
{On effectue les estimations mixtes ayant un comportement suffixe pour le nœud n}
estimation_suf(mix, n)
```

Les phases d'estimation de l'algorithme 1 ne sont pas décrites à dessein. En effet, une version "naïve" de ces phases aurait été un simple parcours de la liste des estimateurs et l'appel de leur fonction d'estimation sur le nœud n. La version "naïve" de l'algorithme de la fonction estimation serait l'algorithme 2.

Algorithme 2 Algorithme naïf d'estimation des valeurs d'un nœud à partir d'un liste d'estimateurs: la fonction *estimation*

ENTRÉES: n un nœud, l la liste des estimateurs

```
pour tout e ∈ l faire
  enregistrer(e.calculerEstimation(n))
fin pour
```

Pourtant l'ordre des estimations dans ce cas est aléatoire et en aucun cas ne garantit que les estimateurs dont d'autres estimateurs sont dépendants seront traités en premier. Il est donc nécessaire de déterminer un ordre pour les estimateurs.

Précédence

Supposons que l'estimateur A est dépendant de l'estimateur B. Dans ce cas, il existe une relation de précédence entre A et B : B doit intervenir avant A. De même si B est dépendant de l'estimateur C, il existe une relation de précédence entre B et C ainsi qu'entre A et C : C doit intervenir avant B donc aussi avant A.

Le concept de précédence est héritée de l'architecture logique des ordinateurs que nous appliquons à des estimateurs plutôt qu'à des processus.

> *Définition.* Une **relation de précédence**, notée $\prec$, sur l'ensemble des estimateurs E est une relation vérifiant :
>
> – $\forall$ e $\in$ E, on n'a pas e $\prec$ e
>
> – $\forall$ e $\in$ E et $\forall$ e' $\in$ E, on n'a pas simultanément e $\prec$ e' et e' $\prec$ e
>
> – la relation $\prec$ est transitive

Dans l'exemple précédent, les relations entre A, B et C sont donc notées :

– $B \prec A$

– $C \prec B$

– $C \prec A$

La relation de précédence peut être représentée sous la forme d'un graphe orienté. La partie gauche de la figure II.7 représente le cas de quatre estimateurs A, B, C et D, l'estimateur C étant dépendant de l'estimateur B. En plus de la relation, nous introduisons le niveau de précédence.

> *Définition.* Le **niveau de précédence** d'un estimateur est la longueur du chemin le plus long parmi ceux le reliant aux estimateurs qu'il précède.

La partie droite de la figure II.7 montre le graphe de précédence ainsi que les niveaux de précédence après l'ajout de l'estimateur E dépendant de A et C. Les niveaux de précédence des estimateurs A, B et C ont augmenté alors que le niveau de D est resté le même.

La mise à jour du graphe de précédence lors de l'ajout d'un estimateur est illustré par l'algorithme 3.

Pour compléter l'algorithme 3, il est nécessaire d'introduire l'algorithme 4 permettant la mise à jour des niveaux de précédence.

Ainsi, grâce au graphe de précédence et aux niveaux de précédence, il est possible de donner un ordre à l'application des estimateurs. En effet les estimateurs doivent être appliqués dans l'ordre décroissant de leur niveau de précédence.

En reprenant l'exemple précédent, imaginons que les estimateurs A, B et D sont préfixes alors que les estimateurs C et E sont suffixes. Dans ce cas, pour un nœud donné,

Algorithme 3 Algorithme d'ajout d'un estimateur au graphe de précédence

ENTRÉES: e un estimateur, g le graphe de précédence

g.ajouter(e, 0) {on ajoute l'estimateur au niveau 0 de précédence}
si e est dépendant **alors**
  pour tout $d \prec e$ **faire**
    créerArc(d, e) {on crée un arc de l'estimateur d à l'estimateur e}
    changerNiveau(d, 1) {on met les niveaux des prédécesseurs de e à 1 s'ils sont inférieurs à 1}
  fin pour
finsi

Algorithme 4 Algorithme de changement de niveau de précédence d'un estimateur : changerNiveau

ENTRÉES: e un estimateur, n le niveau induit par le changement de niveau d'un successeur

$l \leftarrow$ niveau(e) {on prend le niveau actuel de e}
{si le niveau de l'estimateur n'est pas assez haut}
si $l < n$ **alors**
  niveau(e) $\leftarrow n$ {on change le niveau}
  {on répercute le changement de niveau sur les prédécesseurs de l'estimateur e}
  pour tout $d \prec e$ **faire**
    changerNiveau(d, $n + 1$)
  fin pour
finsi

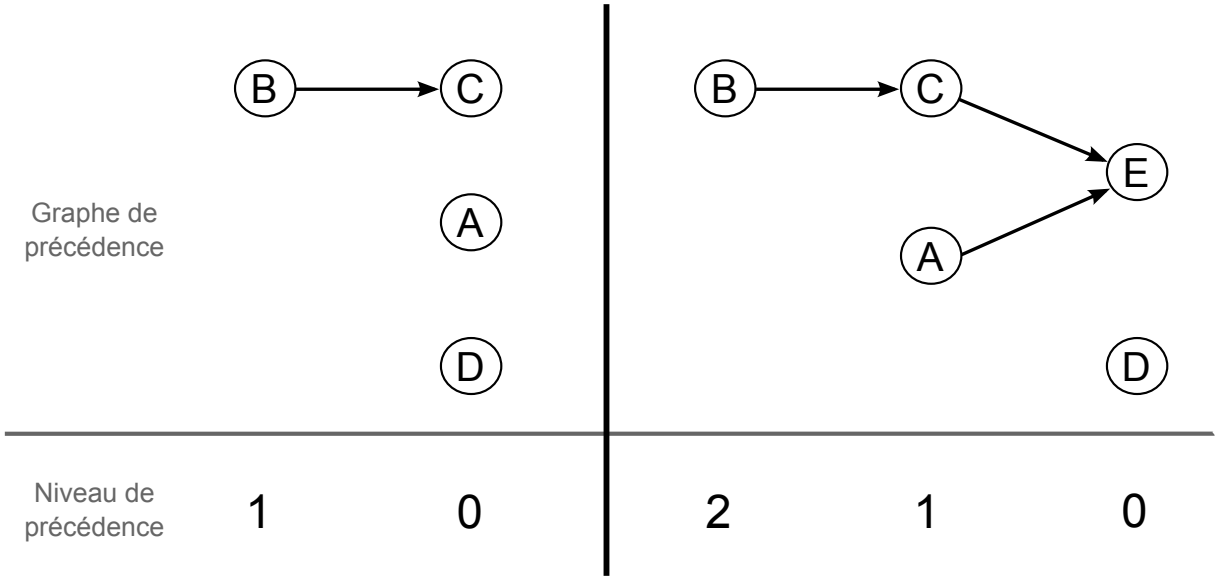

FIG. II.7 – *Graphe de précédence avant (à gauche) et après (à droite) l'ajout de l'estimateur E dépendant de A et C.*

les estimateurs préfixes doivent se faire dans l'ordre $B \rightarrow A \rightarrow D$ et les estimateurs suffixes dans l'ordre $C \rightarrow E$.

Grille d'estimations

La grille d'estimations est le moyen de stocker toutes les estimations. Elle consiste en un tableau à deux dimensions indexé par les nœuds et les estimateurs. Ainsi, avec un nœud et un estimateur, il est possible de retrouver rapidement l'estimation correspondante.

La grille d'estimations est aussi bien utilisée pour écrire les résultats des estimateurs que pour donner les estimations nécessaires aux autres estimateurs. C'est sur cette grille que sont lues les valeurs calculées par les estimateurs prédécesseurs ou par le même estimateur, mais pour le nœud père ou les nœuds fils.

Comme nous le verrons dans la partie II.4, cette grille est aussi bien le pivot de la simulation des transformations que le guide pour leur application.

Un exemple de grille d'estimation est donné dans la figure II.8. Cette grille d'estimation correspond au document Web dont le code est en annexe B et regroupe les estimations présentées plus haut.

	Profondeur du nœud	Nombre de mots	Propriétés horizontales
<body>	0	4	largeur : 640px
#texte	1	4	largeur minimal : 55px largeur maximal : 240px
<a>	1	0	largeur : 150px marge à gauche : 20px
<img>	2	0	largeur : 150px

FIG. II.8 – *Grille d'estimation pour l'arbre DOM.*

II.3.2 Continuité entre les sous-pages

En plus de factoriser les processus nécessaires pour toutes les sous-pages potentielles d'un document Web, le module de session a pour but de garder en mémoire chaque sous-page et son adaptation.

Prenons le cas d'un document Web dont l'adaptation comprend le découpage du document Web en sous-pages. Comme expliqué dans la partie II.4, cette séparation en sous-pages se fait en taillant certaines branches de l'arbre DOM. Ainsi, les sous-arbres qui sont détachés de l' arbre principal doivent être gardés en mémoire.

Les méta-heuristiques n'étant pas des algorithmes déterministes, il faut garder une trace des sous-arbres pour pouvoir adapter exactement les branches manquantes.

De même, afin de ne pas recalculer l'adaptation à chaque requête d'une sous-page, le système enregistre l'adaptation sélectionnée par le moteur de transformation. Ainsi, si la sous-page a déjà été visitée et donc adaptée, le module de session est capable de créer directement la sous-page adaptée sans faire appel au moteur de transformation.

II.3.3 Bilan

Le module de session a un rôle de support pour le moteur de transformation. Il est le garant de la continuité de navigation si un document Web est découpé en plusieurs sous-pages lors de l'adaptation. C'est aussi dans ce module que le traitement préliminaire du document Web est exécuté : la création de l'arbre DOM et l'estimation initiale des valeurs sur l'ensemble de l'arbre. De plus, les estimateurs et les transformations sont enregistrés au niveau du module de session.

La prochaine partie est consacrée au moteur de transformation. Elle explique le déroulement de la sélection de la meilleure adaptation du document Web selon le contexte.

II.4 Moteur de transformation

Le moteur de transformation a pour objet de déterminer les transformations à utiliser et, le cas échéant, leur ordre. Les transformations auxquelles nous nous intéressons sont celles enregistrées lors de la création du module de session. En effet, les transformations, les estimateurs, ainsi que la grille d'estimations sont transmis au moteur de transformation pour déterminer une composition de transformations optimale.

Dans cette section, une première partie introduit le concept de transformation, et une seconde partie s'intéresse à la recherche de la composition optimale.

II.4.1 Transformations

Avant de développer le mécanisme de recherche de la composition optimale, il est nécessaire de définir ce qu'est une transformation dans nos travaux.

Le point de départ de nos transformations est le concept de technique de transcodage. Comme toute technique de transcodage, une transformation doit être atomique et composable. Le terme atomique est pris dans le sens éthymologique : insécable.

Comme nous nous plaçons au niveau de la représentation DOM du document Web, chaque transformation doit pouvoir s'appliquer sur l'arbre DOM plutôt que directement sur le code du document. En cela, notre définition de transformation suit encore le concept de technique de transcodage. Pourtant notre définition s'éloigne des techniques de transcodage par plusieurs points :

- chaque transformation doit pouvoir simuler son application à travers la modification de valeurs dans la grille d'estimation,
- une transformation peut avoir deux portées d'application : globale ou ciblée (expliqué par la suite) et
- chacune des transformations présente un indice de performance dénotant le changement qu'elle provoque dans la cohésion visuelle du document.

Simulation plutôt qu'application directe

Le but du moteur de transformation est, lors de la recherche d'optimum, de déterminer la meilleure composition mais en aucun cas d'appliquer cette composition. L'application des transformations est déléguée au module de session après qu'une composition des transformations soit établie. Puisqu'il n'est pas nécessaire d'appliquer les transformations au niveau du moteur de transformation, une solution moins coûteuse en temps et en puissance de calcul est la simulation des transformations afin de déterminer leur impact dans le cadre d'une composition (mesure de leur pertinence).

Afin d'effectuer ces simulations, chaque transformation doit avoir la capacité de lire et de modifier la grille d'estimations remplie lors de la création du module de session. Ces mécanismes sont regroupés dans la fonction de simulation de la transformation.

Prenons l'exemple d'une transformation qui transforme le texte d'un document Web en parole. On suppose que cette transformation a été développée de façon à ne mettre en audio que les textes de plus de cinquante mots. De plus, la transformation substitue au texte un widget de lecture de piste audio d'une largeur de 100 pixels. Il est clair que cette transformation nécessite l'estimateur du nombre de mots contenus par les nœuds. Dans le cas contraire, il serait difficile de déterminer si la condition d'application de la transformation est vérifiée.

Plaçons-nous dans le cas d'un nœud dont l'estimation du nombre de mots est supérieur à 50. La transformation pourrait s'appliquer, mais n'étant qu'au niveau de la fonction de simulation, les estimations du nombre de mots et de la largeur en pixels pour le nœud considéré vont être modifiées par cette fonction pour avoir comme valeurs 0 et 100 respectivement.

En général, les fonctions de simulation, à l'image des fonctions d'application sont fortement dépendantes de leur transformation. Pourtant, elles s'appuient sur deux listes d'estimateurs :

- une liste d'estimateurs pour calculer la condition de la transformation, que ce soit pour la simulation ou pour l'application,
- une liste d'estimateurs dont les valeurs sont modifiables par la fonction de simulation.

Ce double lien entre les transformations et les estimateurs dévoilé, la vérification des estimateurs dont les transformations sont dépendantes, introduit dans la partie II.3.1, prend tout son sens.

Portées des transformations

Dans le but de terminer la définition des transformations, il est essentiel de séparer les transformations en deux catégories : globale et ciblée. Une transformation globale est susceptible d'être appliquée à n'importe quel nœud de l'arbre DOM et même à plusieurs. Ce type de transformations ne dépend que de sa condition d'application. Dans l'exemple précédent d'une transformation de vocalisation, il est possible que plusieurs nœuds contiennent plus de 50 mots, impliquant ainsi plusieurs simulations de vocalisation.

Au contraire, une transformation dite "ciblée" a un paramètre de plus pour sa fonction de simulation : une liste de nœuds sur lesquels elle doit être simulée. C'est le cas pour la transformation servant à découper un document Web en sous-pages. En effet, plutôt que de limiter l'application de certaines transformations par des conditions figées, nous nous

donnons la possibilité de les appliquer à une liste de nœuds.

Définition complète

Une transformation est un composant logiciel défini par :

- un périmètre d'action : globale ou ciblée,
- une liste d'estimateurs nécessaires pour calculer sa condition d'application,
- une liste d'estimateurs éventuellement modifiés par la fonction de simulation,
- une fonction de simulation capable de modifier des estimations pour refléter l'application possible de cette transformation sur la grille d'estimation,
- une fonction d'application modifiant l'arbre DOM,
- un indice de performance dénotant le changement dans la cohésion visuelle amené par la transformation.

Avant de voir en détail la manière dont le moteur de transformation fonctionne, il est nécessaire de définir une composition de transformations :

> *Définition.* Une **composition de transformations** est une paire de données établissant :
>
> - la liste de nœuds pour chaque transformation ciblée,
> - l'ordre des transformations parmi les transformations globales.

On considère dans notre modèle que les transformations ciblées interviennent avant les transformations globales puisque plus spécifiques.

II.4.2 Algorithme de recherche

Maintenant que le principe de la simulation des transformations a été décrit, nous présentons les travaux permettant de déterminer la composition optimale.

Nous avons pour but de trouver la meilleure composition de transformations étant donnés un document Web et un contexte. Cet espace de recherche dépend :

- du nombre de transformations,
- du nombre de nœuds dans la représentation DOM du document Web et
- du nombre maximal de nœuds qui peuvent être les cibles des transformations ciblées.

Afin d'illustrer la grandeur de l'espace de recherche, partons du postulat que nous traitons un document Web ayant n nœuds avec g le nombre des transformations globales et TC l'ensemble des transformations ciblées. Nous noterons k_i $(i \in TC)$ le nombre maximal de nœuds cibles pour la transformation ciblée d'indice i. Le nombre de possibilités de cibles pour la transformation ciblée d'indice i est donc le nombre de combinaisons avec répétition de 0 à k_i nœuds pris parmi les n nœuds de l'arbre DOM. Etant notés Γ_n^j le nombre de

combinaisons avec répétition de j éléments parmi n et C_k^j le nombre de combinaisons sans répétition de j éléments parmi k, on notera P_i le nombre de possibilités de listes de cibles pour la transformation ciblée i.

$$P_i = \sum_{j=0}^{k_i} \Gamma_n^j = \sum_{j=0}^{k_i} C_{n+j-1}^j = \sum_{j=0}^{k_i} \frac{(n+j-1)!}{j!(n-1)!}$$

Par conséquent, le nombre de possibilités pour les cibles de toutes les transformations ciblées sont P_C :

$$P_C = \prod_{i \in TC} P_i = \prod_{i \in TC} \sum_{j=0}^{k_i} \Gamma_n^j = \prod_{i \in C} \sum_{j=0}^{k_i} C_{n+j-1}^j = \prod_{i \in TC} \sum_{j=0}^{k_i} \frac{(n+j-1)!}{j!(n-1)!}$$

De même, étant noté A_g^j le nombre d'arrangements sans répétition de j éléments parmi g, le nombre d'ordres différents de transformations globales est noté P_G :

$$P_G = \sum_{j=0}^{g} A_g^j = \sum_{j=0}^{g} \frac{g!}{(g-j)!}$$

P_G est donc le nombre d'arrangements sans répétition de 0 à g transformations prises parmi G. Enfin, les solutions possibles à notre problème d'adaptation sont au nombre de S calculé grâce à la formule suivante :

$$S = P_C \times P_G = \left(\prod_{i \in TC} \sum_{j=0}^{k_i} \frac{(n+j-1)!}{j!(n-1)!} \right) \times \left(\sum_{j=0}^{g} \frac{g!}{(g-j)!} \right)$$

Pour se rendre compte de l'explosion du nombre de solutions possibles à notre problème d'adaptation, prenons l'exemple suivant : il n'existe qu'une seule transformation ciblée mais son nombre maximal de nœuds-cibles (k) peut varier. De même, le nombre de transformations globales (g), et le nombre de nœuds dans l'arbre DOM (n) peuvent varier. Le tableau II.1 regroupe les valeur de P_G et P_C ainsi qu'une ordre de grandeur pour le nombre de solutions ($\simeq S$) lors de la variation des dimensions de notre exemple afin d'illustrer la taille de l'espace de recherche sous ces contraintes.

La table II.1 démontre l'explosion de la taille de l'espace de recherche lors de l'augmentation des trois dimensions. Le nombre de transformations globales et le nombre maximal de nœuds-cibles sont des paramètres de notre système. Au contraire, le nombre de nœuds de l'arbre DOM est un paramètre que nous ne pouvons pas maîtriser. Le nombre de transformations globales fait croître rapidement la taille de l'espace de recherche. Pourtant cette dimension appelle à des développements logiciels. Elle est donc peu susceptible de varier très significativement dans le temps. Au contraire, P_C est le plus variable. Sa

$\#G$	k	n	P_G	P_C	$\simeq S$
3	2	10	16	66	1,1e3
6	2	10	1957	66	8,2e4
12	2	10	1302061345	66	8,6e10
3	4	10	16	1717	2,7e4
3	8	10	16	641631	1e7
3	4	20	16	138880	2,6e6
3	5	40	16	2100384	3,4e7

TAB. II.1 – *Variations de P_G, P_C, et $\simeq S$ consécutivement à la variation d'une des trois dimensions de notre exemple.*

valeur est de l'ordre de n^K avec :

$$K = \sum_{i \in C} k_i$$

Donc, plus l'arbre DOM du document contient de nœuds (n augmente), plus la valeur de P_C va s'accroître rapidement. Afin de se représenter l'ordre de P_C, l'arbre DOM d'une page de Wikipedia [14] peut contenir près de 1000 nœuds, ce qui donne un ordre de 1000^K soit 10^{3K}.

Étant donnée la taille de l'espace de recherche, une heuristique ne peut donc seule adresser ce problème de recherche d'optimum. Nous avons choisi d'avoir recours à un algorithme génétique pour trouver une solution approchant l'optimum. Le choix de cette méta-heuristique a été motivée par la robustesse des algorithmes génétiques et la possibilité de jouer sur leurs paramètres comme nous le faisons dans le chapitre III.3.2. Dans la suite de cette partie, nous présentons donc les algorithmes génétiques d'un point de vue généraliste puis nous développons l'implémentation que nous en faisons.

II.4.2.1 Généralités sur les algorithmes génétiques

Les algorithmes génétiques sont des méthodes d'optimisation et de recherche appartenant à la famille des algorithmes évolutionnistes. L'inspiration principale de ces algorithmes vient de la théorie conçue par Charles Darwin en 1838. Plus exactement, les algorithmes génétiques s'appuient sur les principes d'adaptation et d'hérédité de la sélection naturelle.

Le principe général des algorithmes génétiques peut être résumé comme suit : une solution, si elle est adaptée, peut donner naissance à une autre solution, possiblement plus adaptée encore, en lui adjoignant des caractéristiques d'une autre solution.

Ainsi on ne parle plus de solutions mais d'individus, formant une population qui, génération après génération, convergent pour donner des individus de plus en plus adaptés à leur environnement. Ce mécanisme a été formalisée dans la première édition datant de

14. par exemple http://fr.wikipedia.org/wiki/Wiki

1975 du livre de Holland (1992). Il a été ensuite repris et popularisé par D. Goldberg dans Goldberg (1989).

Chaque individu est classiquement codé par un chromosome, reflet de son bagage génétique. Un chromosome est un enchaînement de gènes. Chaque gène a plusieurs versions appelées allèles. La place d'un gène dans un chromosome est appelé locus.

Les phases du principe général des algorithmes génétiquessont :

1. Création de la population de base de manière aléatoire ou guidée.
2. Evaluation des individus à l'aide d'une fonction d'évaluation. L'évaluation d'un individu est le reflet de son adaptation au système.
3. Sélection des individus pour reproduction. Cette sélection s'opère en tenant compte des évaluations mais aussi de la technique de sélection.
4. Croisement des individus sélectionnés pour donner naissance à une nouvelle génération.
5. Mutation éventuelle d'un petit nombre d'individus (avec une probabilité inférieure à un pourcent pour chaque individu).
6. Condition de fin : si la condition de fin est vérifiée, on ne passe pas à la génération suivante. Sinon, le cycle recommence à partir de la phase 2.
7. L'individu retourné par l'algorithme est celui ayant la meilleure évaluation parmi ceux de la dernière génération.

Le principe général des algorithmes étant vu, nous expliquons dans la partie suivante l'implémentation que nous faisons des algorithmes génétiques pour la recherche de la meilleure composition de transformation.

II.4.2.2 Notre modèle pour la recherche de composition de transformations

Codage des individus

Pour l'implémentation d'un algorithme génétique, il est impératif de fixer le codage des individus. Nous avons opté pour un codage à deux chromosomes pour garder la différence entre les transformations globales et ciblées. Un exemple est donné sur la figure II.9. Dans cet exemple, deux transformations ciblées et 7 transformations globales ont été enregistrées au niveau du module de session. Ainsi les cibles de la transformation ciblée TC_1 sont les nœuds N_a à N_d, celles de TC_2 N_α à N_ϵ et l'ordre des transformations globales commence par TG_I pour finir avec TG_{VII}.

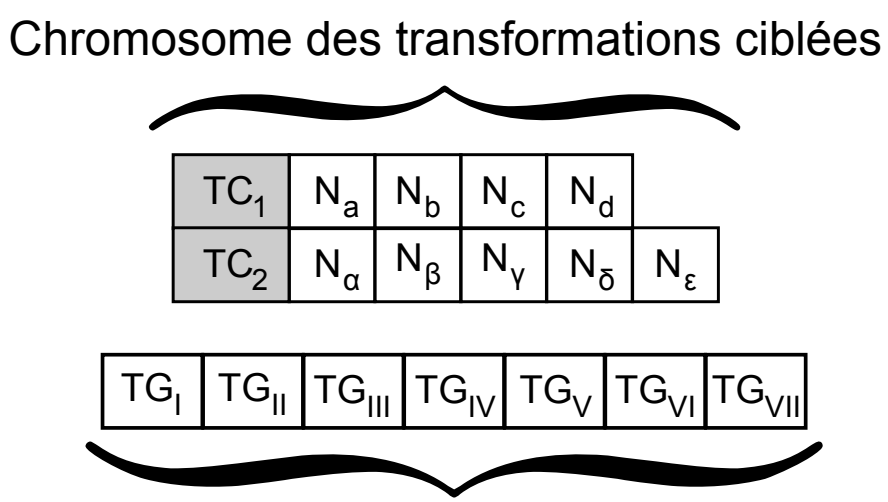

FIG. II.9 – *Génôme d'un individu.*

Création de la première génération

Après avoir fixé le codage des individus, la création de la première génération est le point suivant. Notre choix pour créer cette première population est de créer aléatoirement chaque individu. Cette méthode a l'avantage d'être simple mais surtout de ne pas biaiser le déroulement de l'algorithme avec des individus pré-générés.

Au niveau du chromosome des transformations ciblées, cela se traduit, pour chaque transformation ciblée, par un tirage aléatoire d'un nombre entre 0 et le nombre maximal de cibles note n, puis par un tirage aléatoire de n nœuds en dessous du nœud BODY de l'abre DOM. Ces nœuds constituent les cibles de la transformation.

Quant au chromosome des transformations globales, une transformation globale est tirée au sort puis, dans 1 cas sur 4, est ignorée; sinon elle est ajoutée à la fin de l'ordre. Cette opération est réitérée tant que toutes les transformations globales n'ont pas été traitées.

Croisement et mutation

Les individus dans notre modèle ayant deux chromosomes, la première question à laquelle répondre pour le croisement ainsi que pour la mutation est de savoir si ces mécanismes vont intervenir sur un seul chromosome à la fois ou sur les deux. Nous avons fait le choix de tirer au hasard le chromosome sur lequel est effectué le croisement ou la mutation. Ainsi pour un couple d'individus, leur croisement donne lien à un tirage qui détermine quel chromosome est croisé.

Les contraintes sur les deux chromosomes étant différentes, il est nécessaire d'expliquer séparément chaque mécanisme pour les deux types de chromosomes.

Le croisement pour le chromosome de transformations ciblées consiste à partager les cibles de chaque transformation pour moitié au premier enfant et pour l'autre moitié au second enfant. Ainsi, si les parents ont respectivement 3 et 4 nœuds-cibles pour une transformation donnée, le premier enfant va hériter des 2 premiers nœuds du premier parent et des 2 derniers nœuds du second parent. Au contraire le second enfant aura les 2 premiers nœuds du second parent et le dernier nœud du premier parent.

Le croisement de deux chromosomes de transformations globales est, quant à lui, plus compliqué. En effet, il doit garder l'unicité de chaque transformation globale dans l'ordre. Afin d'illustrer le croisement pour ce chromosome, nous nous appuyons sur la figure II.10.

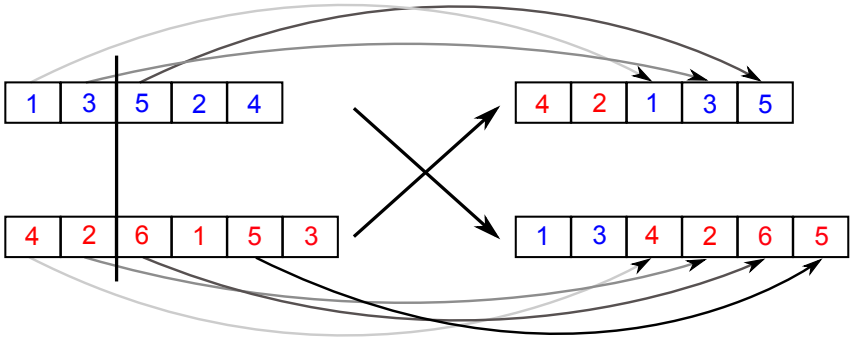

FIG. II.10 – *Croisement d'individus avec locus = 2.*

Dans cette figure, n'apparaîssent que les indices des transformations globales. Les chromosomes à gauche de la figure sont ceux des parents alors que ceux à droite sont ceux des enfants après croisement.

Un locus (valeur entre 1 et la taille du chromosome -1) est tout d'abord tiré aléatoirement. Dans notre exemple, le locus est 2 et nous séparons donc après le gène de locus 2. Ensuite le début de chaque chromosome est recopié sur un enfant, en fait les deux premiers gènes. Enfin, les transformations de l'autre parent sont recopiés dans le même ordre en omettant les transformations héritées du premier parent.

La mutation est d'avantage une affaire de hasard que le croisement. Ainsi, lorsqu'une mutation doit être effectuée sur le chromosome de transformations ciblées, un locus est tiré aléatoirement et le gène à ce locus est remplacé par un nœud pris aléatoirement.

Au contraire, la mutation sur le chromosome des transformations globales n'est pas complètement aléatoire. Une transformation globale est tirée aléatoirement parmi **toutes** les transformations globales et pas seulement celles présentes dans le chromosome. Ensuite, le traitement diffère selon que cette transformation est présente ou non dans le chromosome. Si elle est absente du chromosome, elle est ajoutée à la fin du chromosome, l'ajoutant ainsi en dernier à l'ordre des transformations globales. Si, au contraire, la transformation est déjà présente dans le chromosome, deux cas sont départagés aléatoirement. Le premier cas est la supression du gène correspondant à la transformation. Le second cas, par contre, est le changement de place soit avec le gène précédent, soit avec le gène

suivant dans le chromosome.

Fonction d'évaluation

Comme nous l'avons expliqué auparavant, notre modèle ne vise pas à appliquer systématiquement les transformations mais bien à les simuler pour éviter une perte de temps et de mémoire. L'évaluation est la phase de notre algorithme génétique où la simulation est utilisée.

En effet, comment évaluer chaque individu, c'est-à-dire chaque composition de transformation, par rapport au contexte ? C'est à travers la simulation et donc les estimations modifiées que notre modèle évalue chaque individu.

Le processus d'évaluation des individus est schématisé sur la figure II.11 :

1. Copie de la grille d'estimations. Puisque les simulations vont changer les estimations, il est nécessaire de les appliquer sur une copie de la grille originale.
2. On prend la prochaine transformation, notée T, à traiter dans l'ordre suivant : tout d'abord les transformations ciblées puis les transformations globales dans l'ordre donné par le chromosome correspondant.
3. Si au moins une estimation nécessaire à T a été modifiée auparavant et n'a pas encore déclenché une ré-estimation, alors on passe à l'étape 4, sinon on va directement à l'étape 5.
4. Ré-estimation partielle portant uniquement sur les estimations qui ont été modifiées et pas encore répercutées. Ensuite les estimations modifiées sont remises à zéro.
5. Simulation de la transformation T. La simulation intervient sur les cibles ou sur tout l'arbre selon que la transformation est ciblée ou globale. Les estimations modifiées par T sont ensuite ajoutées aux estimations modifiées.
6. S'il reste des transformations à traiter, alors retour à l'étape 2, sinon passage à l'étape 7.
7. Calcul de l'adaptation de l'individu au contexte.

Si l'étape 5 a été expliquée dans la partie II.4.1, la ré-estimation est un aménagement du processus d'estimation décrit dans la partie II.3.1. En effet, plutôt que d'utiliser tous les estimateurs préfixes, suffixes et mixtes, le processus de ré-estimation ne s'intéresse qu'aux estimateurs dont les estimations ont été modifiées. Ainsi ce ne sont que des sous-ensembles des estimateurs constitués de l'intersection de l'ensemble des estimateurs dont les valeurs ont été modifiées avec l'ensemble des estimateurs utilisés pour la ré-estimation.

Enfin, le calcul de l'adaptation de l'individu au contexte est un aspect crucial de notre algorithme génétique. En effet, ce calcul permet de différencier les critères pris en compte pour juger du meilleur individu, c'est-à-dire de la meilleure composition de transformations. Nous avons retenu quatre critères pour ce calcul : la largeur estimée de la page, la profondeur maximale de l'arbre DOM, le nombre de mots présents sur la page

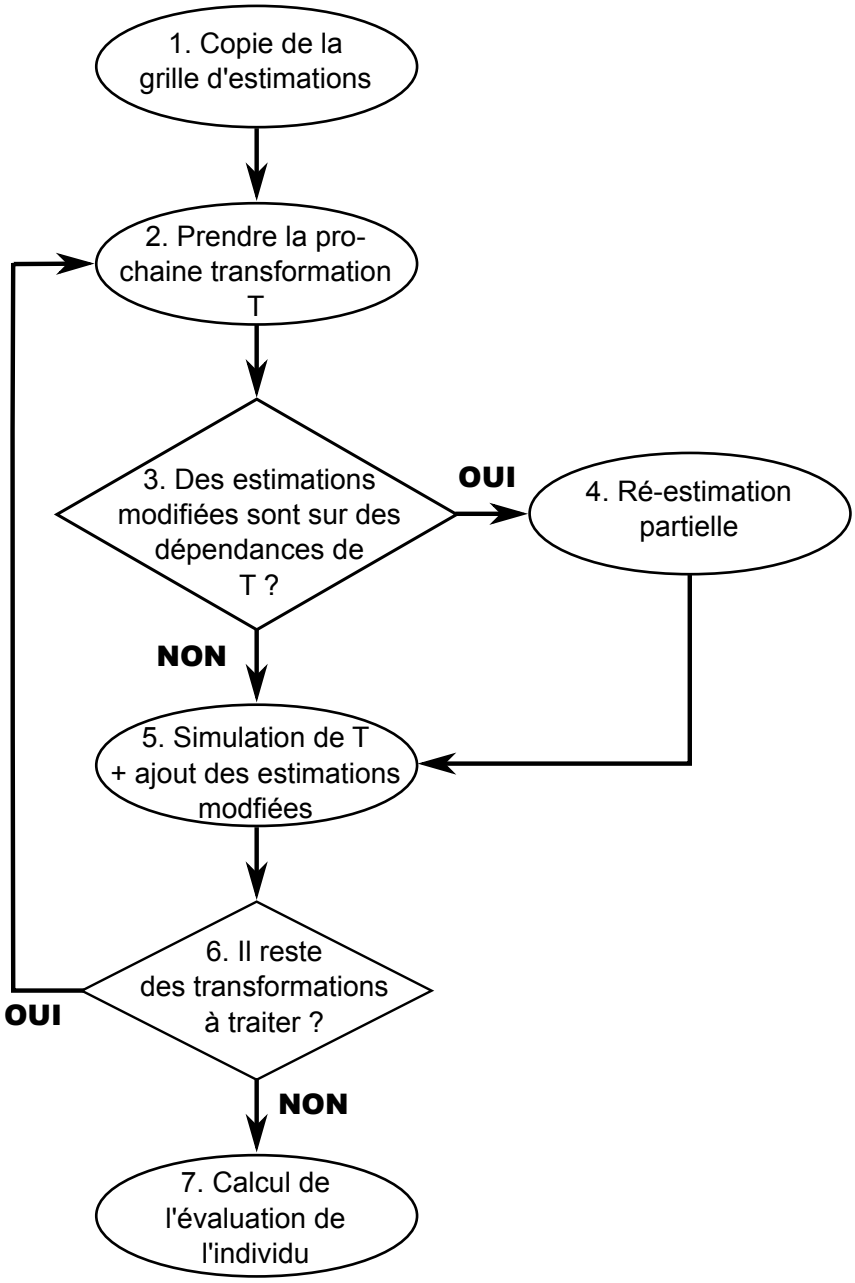

FIG. II.11 – *Processus d'évaluation d'un individu.*

et enfin le coût des transformations en temps et en mémoire. La formule pour calculer l'évaluation E d'un individu est donc :

$$E = \delta_1 \times WR + \delta_2 \times DR + \delta_3 \times WCR + \delta_4 \times PR$$

Où :

- δ_1, δ_2, δ_3 et δ_4 sont les poids modifiables avant l'éxecution (fichier de configuration, cf. partie III.2.1),
- WR est le taux de largeur calculé en divisant la largeur estimée du document adapté par la largeur disponible sur l'équipement de navigation,

- DR dénote le taux de profondeur d'arbre, division de la profondeur de l'arbre DOM après simulation par la profondeur de l'arbre original,
- WCR est le taux de mots, division du nombre de mots dans l'arbre DOM après simulation par le nombre de mots du document original,
- PR est le taux de charge computationnelle calculé par la formule :

$$PR = \frac{TPC - CPC}{TPC}$$

avec :

- TPC la somme des indices de performance de toutes les transformations,
- CPC la somme des indices de performance des transformations effectivement utilisées dans la composition.

Sélection

À partir des méthodes de sélection proposées dans Goldberg & Deb (1991), nous avons opté pour l'étude des différences avec une sélection par tournoi et par roue biaisée sur le rang. La sélection par tournoi consiste à tirer aléatoirement deux individus, de comparer leur évaluation et de sélectionner seulement le meilleur.

Dans la seconde méthode, le rang est calculé comme suit : le dernier en terme d'évaluation obtient le rang 1, l'avant-dernier le rang 2 et ainsi de suite jusqu'au meilleur individu qui obtient un rang égal à la taille de la population. Chaque individu a donc une chance proportionelle à son rang d'être sélectionné.

II.4.3 Bilan

Dans cette partie, nous avons présenté la définition des transformations dans notre modèle et comment un algorithme génétique peut être utilisé pour déterminer la meilleure composition de transformations. Nous avons vu les détails de notre modèle d'algorithme génétique pour la recherche d'un optimum dans l'espace des compositions possibles. De plus les mécanismes d'évaluation vues dans la partie II.3 ont été réutilisés pour l'évaluation des individus lors du déroulement de l'algorithme génétique.

Ainsi s'achève la seconde partie de ce manuscrit. La partie suivante a pour objet la validation scientifique par l'expérimentation de notre modèle.

CHAPITRE II

Modélisation d'une nouvelle approche d'adaptation

Conclusion

Dans ce chapitre, nous avons voulu répondre par un modèle aux enjeux d'universalité et de généricité pour les moteurs d'hypermédia adaptatifs. Nous avons présenté la vue synoptique de notre approche pour ensuite présenter les deux premières contributions de la thèse au domaine, à savoir les concepts de serveur mandataire de contexte et de cookie de contexte trans-domaine.

L'approche générale de notre modèle vise à trouver une suite de transformations à appliquer sur l'arbre DOM d'un document à adapter pour obtenir une adaptation pertinente en regard des contraintes du dispositif pour lequel l'adaptation a lieu. La présentation du module de session est venue mettre en évidence notre formalisation et classification des estimations que l'on peut réaliser à partir d'un arbre DOM. La thèse défend également la possibilité de combiner des estimations des noeuds de l'arbre DOM afin d'obtenir une évaluation plus fine des adaptations envisagées.

Nous avons alors démontré que le problème revient à trouver un optimum parmi un espace de recherche très grand. Ainsi, une approche traditionnelle à base d'heuristiques n'est pas satisfaisante, puisqu'elle reviendrait à tester toutes les combinaisons de transformations possibles de cet espace de recherche. Ainsi, notre modèle intègre une approche par méta-heurtistique. Cela nous permet d'approcher l'optimum en un temps raisonnable (pragmatisme pour viser une implémentation). Parmi les méta-heuristiques, nous avons fait le choix d'une approche évolutionniste et plus particulièrement des algorithmes génétiques. Cet arbitrage correspond à la maturité du domaine des algorithmes génétiques, leur capacité de converger vers une solution proche de l'optimum en un temps raisonnable, ainsi que leur haut degré de paramétrage (notamment via notre fonction d'évaluation comprenant une pondération configurable lors de l'exécution). Nous avons positionné cet algorithme génétique au coeur de notre moteur d'adaptation.

Dans cette thèse, nous nous sommes également attachés à l'implémentation logicielle de notre modèle. C'est une volonté affirmée de notre l'équipe de recherche que de confronter nos modèles à une réalité logicielle. Il va sans dire que l'implémentation de ce modèle n'est pas triviale. Ainsi, le chapitre suivant vise à exposer les détails d'implémentation, et essaie d'évaluer les performances et conditions limites du système.

CHAPITRE III

Implémentations tests et évaluations

Sommaire

CHAPITRE III

Implémentations tests et évaluations

Introduction

Le chapitre précédent eut pour objet notre modèle. Ainsi, nous avons expliqué notre architecture et la distribution du processus d'adaptation entre les trois modules de notre modèle. En outre, nous avons introduit les concepts d'estimateurs et de transformations.

Le présent chapitre vise à éclaircir le lecteur sur les choix technologiques et les détails de l'implémentation que nous avons fait de notre modèle. A travers les librairies que nous utilisons et les estimateurs ou transformations développées, nous présentons le travail effectué pour passer du modèle à une plate-forme de test.

Ensuite, nous expliquons les moyens mis en œuvre pour pouvoir tester notre système en fonction de ses paramètres mais aussi des ressources matérielles mises à disposition.

Enfin, des expérimentations sont conduites pour évaluer notre système. Ces expérimentations portent sur le marcottage qui peut devenir omniprésent et sur la convergence de l'algorithme génétique.

III.1 Implémentation

Cette section a pour objectif de présenter et d'expliquer l'implémentation de notre modèle pour créer une plate-forme de tests. Nous nous sommes attachés au fait d'avoir une application de nos recherches.

Afin de développer notre système, il est nécessaire de faire un choix pour le langage de programmation. La solution retenue a été le langage Java. Ce choix est motivé par plusieurs arguments :

- l'interopérabilité possible de touts les briques entre elles,
- généricité du code par aport aux plates-formes sous-jacentes,
- solidité et pérennité du code,
- notre connaissance approfondie de ce langage de programmation.

Le choix du langage de programmation a bien sûr conditionné le reste des choix technologiques et de l'implémentation. Dans la suite de cette section, nous présentons tour à tour les détails d'implémentation et les choix technologiques pour les trois modules de notre modèle : (1) proxy de contexte, (2) module de session et (3) moteur de transformation.

III.1.1 Implémentation du proxy de contexte

Le proxy de contexte fut le module le plus rapidement implémenté. L'utilisation des librairies HttpCore[1] et HttpClient[2] de la fondation Apache[3] y est pour beaucoup. En effet la librairie HttpCore est utilisée pour la communication avec le client, commençant donc par la réception de la requête HTTP du client et finissant avec la réponse envoyée à celui-ci. Au contraire, la librairie HttpClient permet d'assurer la communication entre le proxy et le serveur Web, rendant possible l'émission de requête au serveur HTTP et l'attente de la réponse de celui-ci.

Pour l'implémentation de la gestion du contexte grâce aux cookies, l'implémentation est la déclinaison technologique du modèle présenté dans les parties II.2.2 et II.2.3 et ne justifie pas une ré-explication complète sous l'angle du langage Java. L'unique limite d'implémentation réside dans le fait que notre modèle laisse la liberté de détection des éléments de contexte, et l'implémentation se borne à la largeur disponible dans le navigateur, la famille et la taille de police par défaut. Cette détection est faite à l'aide du langage EcmaScript.

1. http://hc.apache.org/httpcomponents-core-ga/index.html
2. http://hc.apache.org/httpcomponents-client-ga/index.html
3. http://www.apache.org/

III.1.2 Implémentation du module de session

La création de l'arbre DOM est l'une des tout premières étapes lors de la création d'une session. Notre volonté était déjà dans le modèle de pouvoir accéder aux styles des nœuds de l'arbre DOM de manière transparente. L'analyseur (ou *parser* en anglais) Cobra[4] du projet Lobo[5] est la solution dans notre cas, puisqu'il permet de construire l'arbre DOM d'un document Web mais aussi d'analyser le style du document pour déterminer le style de chaque nœud.

Pour compléter l'implémentation du module de session, il est nécessaire d'aborder les estimateurs implémentées. Pour notre plate-forme de test, nous avons developpé dix estimateurs. Nous les classons en trois catégories : (1) ceux servant à estimer des valeurs en rapport avec le texte, (2) ceux servant à estimer des valeurs en relation avec les propriétés horizontales et (3) celui servant à calculer la profondeur.

Estimateurs en rapport avec le texte

Cette catégorie regroupe les estimateurs développés pour suivre le nombre de mots ou encore si seulement du texte est présent dans une partie de l'arbre DOM.

Les estimateurs de cette catégorie sont au nombre de 5 :

1. *Rien d'autre que texte* (noté RAT) calcule une valeur booléenne renseignant si un élément autre que du texte est présent dans les descendants du nœud concerné.
2. *Comptage des mots* (noté CM) estime le nombre de mots contenus par le nœud concerné ou par ses descendants.
3. *Est dans un lien* (noté EDL) calcule un booléen égal à vrai si le nœud concerné est dans un lien (balise HTML <A> avec un attribut HREF) et faux sinon.
4. *Comptage de mots dans les liens* (noté CML) estime le nombre de mots contenus par le nœud concerné ou par ses descendants aussi contenus dans un lien.
5. *Proportion de mots dans les liens* (noté PML) estime la proportion des mots contenus par le nœud concerné ou par ses descendants aussi contenus dans un lien, par rapport au nombre total de mots.

Estimateurs en relation avec les propriétés horizontales

Les estimateurs de cette catégorie ont été développés afin de supporter l'estimation des propriétés horizontales de chaque nœud de l'arbre DOM. Les propriétés visuelles horizontales d'un nœud sont définies comme les valeurs liées à l'affichage du nœud par un navigateur :

– la largeur ou la largeur minimal et maximal,

4. http://lobobrowser.org/cobra.jsp
5. http://lobobrowser.org/

- la distance à droite et à gauche dans le cas d'un placement absolu,
- les largeurs des bordures droite et gauche,
- les marges horizontales à l'extérieur des bordures,
- les marges horizontales entre les bordures et l'élément.

Cette catégorie compte 4 estimateurs :

1. *Style hérité* (noté SH) regroupe les informations de style héritées des ancêtres du nœud concerné comme la taille ou la couleur de police.
2. *Affichage, position et flottant* (noté APF) renseigne les valeurs finales des propriétés CSS *display*, *position* et *float*. Ces valeurs sont regroupés car elles sont fortement liées selon la spécification CSS 2.1[6].
3. *Largeur dépendante des enfants* (noté LDE) calcule un booléen égal à vrai si la largeur du nœud concerné est dépendant de ces enfants, à l'image d'un tableau HTML dépendant de la taille des cellules.
4. *Formattage horizontal* (noté FH) calcule les estimations des valeurs des propriétés visuelles horizontales.

Estimateur de la profondeur

La dernière catégorie d'estimateur n'en regroupe qu'un seul. C'est l'estimateur de la profondeur (noté PA), intervenant dans l'évaluation des individus dans l'algorithme génétique.

Plutôt que de calculer la profondeur du nœud dans l'arbre, cet estimateur calcule la profondeur du sous-arbre dont le nœud concerné est la racine. L'estimation pour une feuille de l'arbre est donc 0. Par contre, les autres nœuds ont une valeur de profondeur calculée en incrémentant le maximum des estimations de leurs enfants.

Bilan sur les estimateurs

Les estimateurs déjà développés répondent au besoin d'évaluer les individus de l'algorithme génétique et servent de guide aux transformations. Il est intéressant de voir leurs répartitions dans les catégories préfixes, suffixes et mixtes ainsi que les relations de précédence qui les lient.

Parmi les dix estimateurs, 4 sont préfixes :

1. *est dans un lien* (EDL),
2. *style hérité* (SH),
3. *affichage, position et forme* (APF),
4. *largeur dépendante des enfants* (LDE).

6. http://www.w3.org/TR/CSS2/

Cinq estimateurs sont de type suffixe :

1. *rien d'autre que texte* (RAT),
2. *comptage des mots* (CM),
3. *comptage des mots dans des liens* (CML),
4. *proportion de mots dans les liens* (PML),
5. *profondeur de l'arbre* (PA).

Enfin un seul estimateur est mixte : *formattage horizontal* (FH).

La figure III.1 présente le graphe de précédence des estimateurs développés pour notre système.

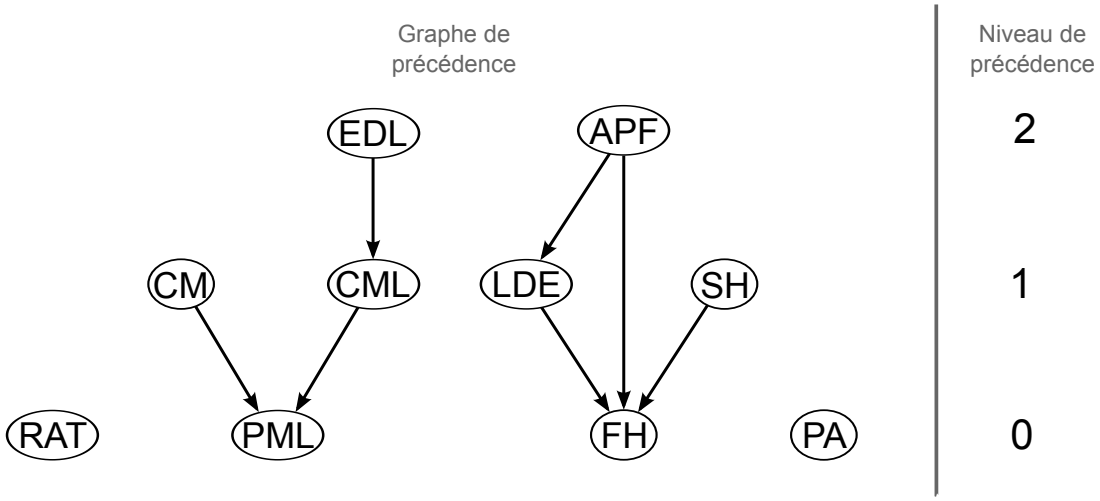

FIG. III.1 – *Graphe de précédence des estimateurs développés avec niveau de précédence.*

III.1.3 Implémentation du moteur de transformation

Le moteur de transformation a été développé sans le concours de librairies bien que certaines existent pour déployer rapidement un algorithme génétique. Ce choix a été fait pour pouvoir maîtriser au maximum chaque étape de l'algorithme. De plus le codage des individus est très spécifique à notre approche des individus (cf. section II.4.2.2). Ceci consistuait donc un obstacle à l'utilisation d'une librairie tiers.

Cette partie se concentre donc particulièrement sur les transformations que nous avons développées. Elles sont au nombre de quatre : trois transformations globales et une transformation ciblée. La première transformation globale est appelée *Texte en voix* (raccourcie en T2V) et correspond typiquement à la lecture d'un paragraphe texte dans un document Web. Pourtant cette vocalisation d'un sous-arbre est limitée par plusieurs conditions :

1. le sous-arbe contient seulement des éléments visuels de type texte (pas d'image, de widget),
2. aucun lien n'est présent dans le sous-arbre et

3. le nombre de mots du sous-arbre est supérieur à un seuil, par défaut 50.

Ensuite, nous avons développé la transformation *Wiki en voix + liens* notée W2V+L. Cette transformation substitue à un contenu de type Wiki une vocalisation assortie d'une liste de liens. Les conditions pour son application sont :

1. le sous-arbre considéré contient seulement des éléments visuels de type texte,
2. des liens sont présents dans le sous-arbre mais l'estimation de *proportion de mots dans les liens* est inférieure à un seuil ,par défaut 0,25, et
3. le nombre de mots du sous-arbre est supérieur à un seuil ,par défaut 50,.

Les figures III.2 et III.3 montrent l'application de la transformation W2V+L sur un sous-arbre d'un document Web de type Wiki.

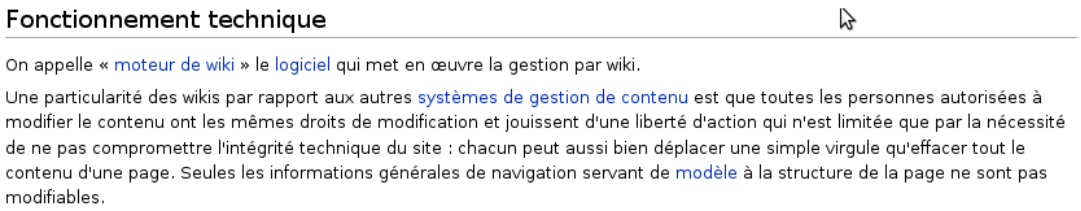

FIG. III.2 – *Affichage d'un sous-arbre sans transformation.*

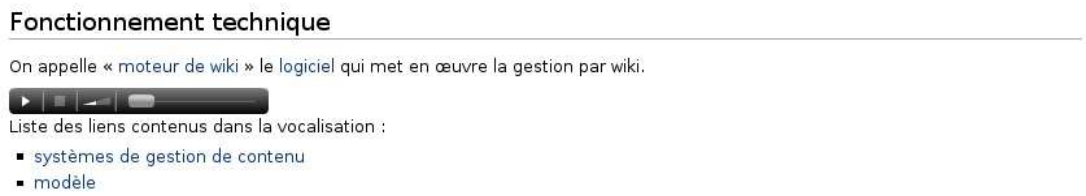

FIG. III.3 – *Affichage d'un sous-arbre après application de la transformation W2V+L.*

Ces deux premières transformations font appel à la synthèse vocale. Afin de rendre cette synthèse possible, nous utilisons la librairie Vocalyze[7] de Polytech'Nice-Sophia[8]. Cette librairie s'appuie sur le synthétiseur vocal Mbrola[9] développée par l'école Polytechnique de Mons[10] (Belgique).

Enfin, la dernière transformation globale est la *mise à l'échelle de la taille de police*. Cette transformation s'applique sur les éléments du document Web dont la taille de police (propriété CSS *font-size*) est absolue (en pixels donc). Elle a pour effet de changer la taille de police en proportion de la taille de police par défaut du contexte. Ainsi la nouvelle taille de police notée T_n est dépendante de celle donnée dans le document (T_d) et celle par défaut du contexte (T_c) :

7. http://vocalyse.polytech.unice.fr
8. http://www.polytechnice.fr/
9. http://tcts.fpms.ac.be/synthesis/
10. http://www.umons.ac.be/

$$T_n = \frac{T_d \times T_c}{16}$$

seize (16) étant la taille de police par défaut sur les navigateurs des ordinateurs de bureau.

La transformation ciblée développée est le *marcottage*. Ce terme venant de la botanique définit l'action de couper une branche d'un végétal pour le replanter et le faire croître indépendemment du végétal d'origine. Cette transformation est la seule capable de découper un document Web en plusieurs sous-pages. Pour cela, le marcottage remplace le sous-arbre dont la racine est une de ses cibles par un lien spécifique. La navigation entre les sous-pages est maintenue par le module de session comme il est expliqué dans la partie II.3.2.

Afin d'observer les liens entre les estimateurs et les transformations, le tableau III.1 résume :

- les estimateurs nécessaires à la vérification des conditions d'application et
- les estimateurs dont les valeurs sont modifiées lors de la simulation.

III.1.4 Bilan

Pour résumer le travail effectué sur notre plate-forme de test, nous avons utilisé 4 librairies en tout pour le développement ; ce qui implique une grande majorité du développement fait par nos soins. Même si un manuscrit de thèse ne permet pas d'en faire état de manière exhaustive, il est important de souligner ici la quantité de développements logiciels menés afin d'obtenir une réalisation logicielle probante de notre modèle. Cela implique le développement de tous les estimateurs, transformations, du codage des individus dans l'algorithme génétique, de l'algorithme génétique lui-même, du module de session, du mécanismes de cookie transdomaine, etc. Ce sont autant de développements logiciels à mettre au crédit de cette thèse. Toutes ces réalisations technologiques ont été implémentées sur des serveurs virtualisés (VMWare[11]) afin de pouvoir observer la performance de l'algorithme (cf. section III.3).

On peut noter que, à des fins de test, que nous avons développé quatre transformations ainsi que les estimateurs nécessaires à ces transformations et l'évaluation des individus de l'algorithme génétique.

Enfin, le fait d'avoir quatre transformations implique des valeurs de G et de C à 3 et 1 respectivement, conformément aux définitions de la partie II.4.2. Par conséquent, P_G a une valeur de 16, ce qui est, pour un espace de recherche, un petit multiplicateur. Mais l'espace de recherche explose en grandeur sous l'effet de P_C. Supposons maintenant

11. http://www.vmware.com/fr/

Transformation	**Estimateur(s) nécessaire(s) pour la condition de la transformation**	**Estimateur(s) dont les valeurs sont modifiées par la transformation lors de la simulation**
Texte en voix	– *comptage des mots* – *comptage des mots dans les liens* – *rien d'autre que texte*	– *comptage des mots* – *rien d'autre que texte* – *profondeur d'arbre* – *formattage horizontal*
Wiki en voix + liens	– *comptage des mots* – *proportion de mots dans les liens* – *rien d'autre que texte*	– *comptage des mots* – *comptage des mots dans les liens* – *rien d'autre que texte* – *profondeur d'arbre* – *formattage horizontal*
Mise à l'échelle de la taille de police	– *style hérité*	– *style hérité*
Marcottage	$\emptyset$	– *comptage des mots* – *comptage des mots dans les liens* – *rien d'autre que texte* – *profondeur d'arbre* – *formattage horizontal*

TAB. III.1 – *Liens entre les estimateurs et les transformations : dépendences et modifications induites.*

qu'au maximum cinq cibles peuvent être marcottés et que l'arbre DOM du document Web à adapter contient 100 nœuds. Dans ce cas, le nombre de possibilités est supérieur à 9 milliards. Ainsi, si une page de Wikipédia est pris comme document à adapter, le nombre de possibilités atteint l'ordre de 1e15 concernant les combinaisons possibles dans l'espace de recherche.

III.2 Mises en place des expérimentations

En plus de préciser l'implémentation faite de notre modèle, il est nécessaire d'avoir un moyen de faire varier les paramètres du système et d'apporter un maximum d'informations sur le matériel servant aux expérimentations. Cela nous a particulièrement permis de confronter différentes configurations de paramétrage de notre algorithme génétique. Ainsi nous présentons notre système de configuration puis la plate-forme matérielle.

III.2.1 Fichier de configuration

Afin de pouvoir faire varier les paramètres d'une exécution à l'autre, il était nécessaire de centraliser ces paramètres. C'est ainsi que nous avons mis en place un fichier de configuration. Ce fichier regroupe les paramètres dont les principaux sont listés dans la table III.2.

Un exemple de notre fichier de configuration est disponible en annexe C pour le lecteur intéressé.

Paramètre configurable	**Valeurs possibles**
Chemin du dossier de journalisation	– un chemin valide vers un dossier
Journalisation des estimations	– *oui* – *non*
Malus de performance pour chaque transformation	– un entier positif, dont une valeur élevée indique un impact important sur la cohésion visuelle
Nombre maximal de cibles pour le marcottage	– un entier positif voire nul
Journalisation de la session	– 0 : aucune journalisation – 1 : journalisation dans le fichier session.log – 2 : journalisation en console
Taille de la population de l'algorithme génétique	– un entier positif

Nombre de générations pour l'algorithme génétique	– un entier positif
Configuration du croisement	– vrai : croisement élitiste – faux : croisement normal
Type de sélection	– vrai : par roue biaisée sur le rang – faux : par tournoi
Taux de croisement	– un nombre décimal entre 0 (exclus) et 1
Taux de mutation	– un nombre décimal entre 0 (exclus) et 1
Journalisation de l'algorithme génétique génération après génération	– 0 : aucune journalisation – 1 : journalisation des évaluations-types : meilleure, pire, moyenne, médiane – 2 : idem que 1 + journalisation des individus-types : meilleur, pire, médian – 3 : idem que 2 + création des pages correspondantes aux meilleurs individus – 4 : idem que 3 + création des pages correspondantes aux pires individus
Nombre de tâches allouées à l'évaluation des individus	– un entier positif, idéalement inférieur au nombre de processeurs logiques disponibles
Pondérations des critères d'évaluation des individus	– un nombre décimal entre 0 et 1 dont la somme est idéalement égale à 1

TAB. III.2 – Liens entre les estimateurs et les transformations : dépendences et modifications induites.

Ainsi, pour les expérimentations, nous serons capables d'étudier l'influence de chacun des paramètres du tableau III.2.

III.2.2 Serveur de test

Bien que le fichier de configuration soit un moyen efficace de changer les valeurs de notre système, ce système ne permet pas de changer des valeurs intrinsèques à la couche matérielle. Ainsi, nous nous sommes penchés sur les possibilités de faire varier les paramètres matériels tels que le nombre de processeurs physiques, la mémoire vive allouée. C'est ainsi que nous avons décidé d'utiliser un serveur de virtualisation. Cette technologie permet de déployer sur un même serveur physique plusieurs machines virtuelles dont les éxecutions sont indépendantes.

La virtualisation a donc pour avantage de rendre possible la conduite de plusieurs expérimentations en parallèle. De plus, chaque machine virtuelle est configurable en terme de ressources matérielles.

Les caractéristiques de notre serveur de virtualisation sont regroupées dans le tableau III.3.

Modèle	Dell PowerEdge 1950
Version du BIOS	2.3.1
Processeur	Intel®Xeon®
Vitesse de processeur	2,7 GHz
Sockets de processeurs	2
Processeurs par socket	4
Processeurs logiques	8
Hyperthreading	N/A
Mémoire vive (RAM)	8 Go
Mémoire vive allouée pour le système de virtualisation	821,6 Mo
Mémoire vive disponible pour les machines virtuelles	7362 Mo
Disque dur	372 Go
Disque dur alloué pour le système de virtualisation	5 Go
Disque dur disponible pour les machines virtuelles	367 Go
Logiciel de virtualisation	ESXi 4 Single Server licensed for 2 physical CPU

TAB. III.3 – *Caractéristiques techniques de notre serveur de virtualisation*

S'il est possible de configurer le nombre de processeurs lors de la création de la machine virtuelle, il est déconseillé de le faire varier une fois que le système d'exploitation est installé. De plus, nous sommes limités à quatre processeurs sur une même machine virtuelle. Cette restriction vient de la license du logiciel de virtualisation. Ces deux facteurs font que nous avons créé quatre machines virtuelles identiques, dont seul le nombre

de processeurs diffère (1, 2, 3 ou 4).

Afin de comparer les expérimentations, il était nécessaire que chaque machine virtuelle soit sur le même pied d'égalité. C'est ainsi que le même système d'expérimentation a été installé sur les machines virtuelles. Notre choix s'est porté sur la version serveur d'Ubuntu 9.10. De plus, les paquets Linux installés sont rigouresement les mêmes et dans la même version sur chaque machine virtuelle. Ainsi, les différences entre les expérimentations sur chaque machine virtuelle ne peuvent être imputées à une différence de configuration du système d'exploitation.

III.2.3 Bilan

À des fins d'expérimentations, nous avons, dans un premier temps, mis en place un système de configuration. Il a pour but de nous donner la capacité de changer les valeurs des paramètres du système. Pourtant, les paramètres matériels ne pouvaient être modifiées par ce moyen. La solution à ce problème s'est avérée être le déploiement d'un serveur de virtualisation et de quatre machines virtuelles. Ainsi, nous avons la possibilité de faire varier autant les paramètres de notre plate-forme que ceux matérielles. La partie suivante sera donc dédiée à l'expérimentation.

III.3 Expérimentations et résultats

Nous proposons dans cette section la validation de notre modèle à travers une évaluation de notre plate-forme de tests. La première partie de cette section est dévolue à l'étude du phénomène que nous avons nommé sur-marcottage. Ensuite, nous étudions la convergence de l'algorithme génétique et quels sont les paramètres du système qui rentrent le plus en compte dans cette convergence.

III.3.1 Du danger du sur-marcottage

Lors des premières expérimentations, nous avons découvert un comportement inattendu pour notre plate-forme de test. En effet, il s'avérait que la composition optimale pour certains documents Web était toujours un marcottage unique du fils du nœud BODY sans qu'aucune autre transformation ne soit utilisée.

Ainsi, l'adaptation du document donnait comme résultat un grand nombre de sous-pages ne contenant que quelques nœuds. A force d'investigation, nous avons découvert que ce comportement était dû à la configuration des pondérations des critères de l'évaluation des individus dans l'algorithme génétique. Une pondération favorisant le taux de profondeur de l'arbre et le taux de mots favorise un marcottage à outrance.

C'est ainsi qu'il nous parut important de parler de ce défaut possible dans la configuration du système. La solution que nous avons découvert pour pallier ce problème est la suivante :

- équilibrer les pondérations des taux de profondeur d'arbre et de mots d'un côté et la pondération du taux de charge computationnelle de l'autre et
- surdimensionner le malus de performance du marcottage par rapport aux malus des autres transformations.

Ainsi, plutôt que de privilégier une adaptation par marcottage à granularité fine, le système privilégiera une utilisation balancée des transformations.

III.3.2 Étude de la convergence de l'algorithme génétique

Le paramétrage de ce premier cas d'étude a été le suivant :

- cinq cents (500) individus dans la population,
- deux cent (200) générations,
- la transformation de marcottage est limitée à cinq (5) noeuds au maximum,
- le pourcentage de mutation est de 0,1%,
- des pondérations en faveur des taux de largeur et de mots.

Nous nous sommes plus particulièrement intéressés à l'adaptation d'un document Web de type wiki, avec un nombre de nœuds avoisinant les mille (1000) éléments. Le contexte pour lequel nous avons adapté le document est celui d'un téléviseur utilisant une SetTop Box (640 pixels de résolution horizontale et une taille de police par défaut de 24 pixels. La figure III.4 montre l'affichage de notre document sur un ordinateur de bureau mais avec l'équivalent des paramètres de l'affichage sur un téléviseur.

FIG. III.4 – *Aperçu de l'affichage d'un document de type Wiki sur un téléviseur.*

Cette expérimentation a pour but d'étudier la convergence de l'algorithme génétique dans le cas où le marcottage d'un nœud ou d'un petit nombre de nœuds prédominait lors de l'évaluation de l'individu.

Dans notre cas, les pondérations sont déséquilibrées, comme dans l'explication de la partie précédente. Ceci implique que l'adaptation optimale se résume au marcottage du seul fils du nœud BODY. Ceci reflète bien la convergence de manière générale pour notre système. En effet, les transformations globales étant en nombre limité et ce nombre risquant de ne pas grandement varier, la taille de l'espace de recherche est principalement dépendante du nombre de nœuds dans l'arbre DOM.

Avant de mener notre expérimentation, nous avions supposé que la faculté de l'algorithme à converger vers un optimum dans un cas pareil devait être liée à un ou plusieurs

des paramètres suivants :

- la méthode de croisement (normal ou élitiste),
- le mode de sélection des individus (par rang ou par tournoi),
- à la taille de la population,
- au nombre de générations.

De plus, nous nous interrogions sur l'utilité de l'écart-type entre les évaluations comme condition d'arrêt de l'algorithme génétique

Afin de mettre le maximum de chance de notre côté, nous avons décidé de prendre pour notre première expérimentation une taille de population (500) proche de la moitié du nombre de nœuds de l'arbre DOM du document Web et un nombre conséquent de générations (200). De plus, nous avons voulu tester l'influence de la méthode de croisement et du mode de sélection sur plusieurs exécutions. C'est la raison pour laquelle nous avons mené dix (10) exécutions de l'adaptation de la première sous-page pour chaque couple de valeurs du mode de sélection et de la méthode de croisement.

Les tableaux III.4 et III.5 regroupent les résultats des quarantes (40) exécutions de l'adaptation.

Il est à noter que les numéros de génération commencent à 0 et finissent à 199 pour ces exécutions.

Ces tableaux présentent en surligné vert les exécutions dont le dernier individu est l'optimum global, ce qui implique que l'optimum global est l'individu sélectionné pour quinze (15) des quarante (40) exécutions.

Nous observons aussi que quatre exécutions n'ont pas terminées avec la meilleure individu qui se soit présenté au fil des générations. Ceci s'explique par le fait que ces exécutions utilisent une méthode de croisement non-élitiste, ne garantissant donc pas que les meilleurs individus fassent partie de la génération suivante.

On peut noter que, pour trente-cinq (35) exécutions (donc toutes à l'exception des exécutions 2, 9, 25, 27 et 28), lorsqu'un optimum est trouvé, même s'il est local, l'algorithme génétique stagne à cet optimum jusqu'à la fin de l'algorithme. Cet état de fait est facilement déterminé en auditionnant les cinquième et sixième valeur d'une ligne.

Quant à l'écart-type, il ne nous renseigne pas plus sur la convergence ou non vers l'optimum local. En effet, lors de deux exécutions, l'optimum global a été trouvé et pourtant pas un seul écart-type est inférieur à 1E-12. De plus, l'écart-type peut être très petit et ceci pendant plus de la moitié des générations alors que l'algorithme stagne dans un optimum local comme c'est le cas pour les exécutions 1, 4 ou 20.

Enfin, huit exécutions avec une méthode de croisement non-élitiste et cinq avec un croisement élitiste ont fini avec l'optimum global. En ce qui concerne le mode de sélection, 6 exécutions utilisant la sélection par rang et 9 utilisant la sélection par tournoi ont

Numéro de l'exécution	Méthode de croisement (normale ou élitiste)	Mode de sélection (rang ou tour-noi)	Meilleure évaluation (E_b)	Première génération de la meilleure évaluation (G_b)	Nombre de générations avec la meilleure évaluation	Écart-type à la génération G_b	Génération du premier écart-type inférieur à $1E-12$	Nombre de générations avec un écart-type inférieur à $1E-12$	Nombre de générations ayant un écart-type inférieur à $1E-12$ et présentant la meilleure évaluation	L'algorithme finit-il avec la meilleure évaluation rencontrée lors de l'expérimentation?
1	normale	rang	0,9158	11	189	1,9218	22	118	118	oui
2	normale	rang	0,9613	4	195	2,4421	16	143	143	oui
3	normale	rang	0,8623	17	183	0,8545	27	108	108	oui
4	normale	rang	0,9613	6	194	3,0851	12	138	138	oui
5	normale	rang	0,8750	14	186	1,5922	25	107	107	oui
6	normale	rang	0,9613	6	194	2,8927	16	150	150	oui
7	normale	rang	0,9613	15	185	0,0124	14	121	120	oui
8	normale	rang	0,8869	65	135	0,0317	28	90	71	oui
9	normale	rang	0,8721	9	5	1,7641	24	109	0	non
10	normale	rang	0,9158	27	173	0,0125	23	110	109	oui
11	élitiste	rang	0,8904	78	122	0,0021	72	63	60	oui
12	élitiste	rang	0,8769	13	187	0,0384	24	98	98	oui
13	élitiste	rang	0,8975	9	191	1,2468	N/A	0	0	oui
14	élitiste	rang	0,9613	6	194	3.1782	25	1	1	oui
15	élitiste	rang	0,8637	28	172	0,4082	39	103	103	oui
16	élitiste	rang	0,9613	7	193	1,8274	17	135	135	oui
17	élitiste	rang	0,9313	9	191	1,6954	19	120	120	oui
18	élitiste	rang	0,9413	87	103	0,0710	23	91	54	oui
19	élitiste	rang	0,9613	4	196	2,3579	121	1	1	oui
20	élitiste	rang	0,9513	12	188	1,5825	24	114	114	oui

TAB. III.4 – *Résultats des 20 premières expérimentations sur la convergence de l'algorithme génétique*

trouvé l'optimum global. Cependant, sur une quarantaine d'exécutions, ces valeurs ne garantissent en rien des résultats similaires pour une autre expérimentation. Ces premiers résultats seront à confirmés avec la réalisation d'un plus grand nombre de tests.

C'est pour cela que nous avons mené une autre expérimentation sur la convergence de l'algorithme génétique en gardant les mêmes paramètres à l'exception de la taille de population passée à 1000 et du nombre de générations diminué à 50. Les tableaux III.6 et III.7 présentent les résultats de cette expérimentation.

Numéro de l'exécution	Méthode de croisement (normale ou élitiste)	Mode de sélection (rang ou tournoi)	Meilleure évaluation (E_b)	Première génération de la meilleure évaluation (G_b)	Nombre de générations avec la meilleure évaluation	Écart-type à la génération G_b	Génération du premier écart-type inférieur à $1E-12$	Nombre de générations avec un écart-type inférieur à $1E-12$	Nombre de générations ayant un écart-type inférieur à $1E-12$ et présentant la meilleure évaluation	L'algorithme finit-il avec la meilleure évaluation rencontrée lors de l'expérimentation?
21	normale	tournoi	0,9613	3	197	1,5727	12	149	149	oui
22	normale	tournoi	0,8848	34	166	0,2852	45	93	93	oui
23	normale	tournoi	0,9613	8	192	3,0623	18	131	131	oui
24	normale	tournoi	0,0,9613	37	163	0,0125	12	134	115	oui
25	normale	tournoi	0,8840	16	2	0,3965	19	105	0	non
26	normale	tournoi	0,9613	14	186	0,0140	13	130	129	oui
27	normale	tournoi	0,9425	0	1	1,4566	22	109	0	non
28	normale	tournoi	0,9425	1	1	1,2446	55	86	0	non
29	normale	tournoi	0,8856	15	185	1,6615	26	106	106	oui
30	normale	tournoi	0,9613	8	192	2,8408	18	130	130	oui
31	élitiste	tournoi	0,8942	7	193	2,0509	15	100	100	oui
32	élitiste	tournoi	0,9513	10	190	1,3792	N/A	O	O	oui
33	élitiste	tournoi	0,9513	8	192	2,5174	19	115	115	oui
34	élitiste	tournoi	0,8983	79	121	0.0033	26	101	64	oui
35	élitiste	tournoi	0,9613	4	196	2,2406	N/A	0	0	oui
36	élitiste	tournoi	0,9613	3	197	1,5048	N/A	0	0	oui
37	élitiste	tournoi	0,9613	9	191	2,0611	16	145	145	oui
38	élitiste	tournoi	0,8813	21	179	1,3463	99	75	75	oui
39	élitiste	tournoi	0,9612	6	194	2,8042	103	1	1	oui
40	élitiste	tournoi	0,9167	158	42	1,1412	190	1	1	oui

TAB. III.5 – *Résultats des 20 dernières expérimentations sur la convergence de l'algorithme génétique*

Cette seconde expérimentation confirme le peu d'utilité de l'écart-type pour détecter l'optimum global puisque, cette fois, ce sont cinq exécutions qui ne présentent pas d'écart-type inférieur à 1E-12 alors qu'elles ont trouvées l'optimum global.

Au contraire, l'augmentation de la taille de la population semble être bénéfique à la convergence vers l'optimum global car, pour cette expérimentation, les exécutions ayant trouvé l'optimum global sont au nombre de vingt-sept (27).

Numéro de l'exécution	Méthode de croisement (normale ou élitiste)	Mode de sélection (rang ou tournoi)	Meilleure évaluation (E_b)	Première génération de la meilleure évaluation (G_b)	Nombre de générations avec la meilleure évaluation	Écart-type à la génération G_b	Génération du premier écart-type inférieur à $1E-12$	Nombre de générations avec un écart-type inférieur à $1E-12$	Nombre de générations ayant un écart-type inférieur à $1E-12$ et présentant la meilleure évaluation	L'algorithme finit-il avec la meilleure évaluation rencontrée lors de l'expérimentation?
1	normale	rang	0,9313	8	42	3,9354	22	14	14	oui
2	normale	rang	0,9613	2	45	1,2894	18	22	22	oui
3	normale	rang	0,9613	2	48	1,9588	N/A	0	0	oui
4	normale	rang	0,8918	18	32	2,1157	31	6	6	oui
5	normale	rang	0,9613	5	45	2,9194	16	16	16	oui
6	normale	rang	0,9613	8	42	2,8615	18	15	15	oui
7	normale	rang	0,9613	12	38	0,6601	21	18	18	oui
8	normale	rang	0,9613	2	48	1,4701	16	19	19	oui
9	normale	rang	0,9613	7	43	4,4752	17	18	18	oui
10	normale	tournoi	0,9613	1	48	1,9540	15	20	20	oui
11	élitiste	rang	0,9613	6	44	3,4033	N/A	0	0	oui
12	élitiste	rang	0,9413	10	40	2,6012	27	5	5	oui
13	élitiste	rang	0,961	5	45	3,1438	13	25	25	oui
14	élitiste	rang	0,8845	19	31	0,8158	29	15	15	oui
15	élitiste	rang	0,9413	14	36	1,5332	N/A	0	0	oui
16	élitiste	rang	0,9513	18	32	1,5032	31	8	8	oui
17	élitiste	rang	0,9613	6	44	3,4651	13	17	17	oui
18	élitiste	rang	0,9613	7	43	3,1022	15	24	24	oui
19	élitiste	rang	0,8915	21	29	0,9838	32	12	12	oui
20	élitiste	rang	0,9613	13	37	0,6236	22	14	14	oui

TAB. III.6 – *Résultats des 20 premières expérimentations sur la convergence de l'algorithme génétique*

Enfin, les exécutions dont la méthode de croisement est élitiste et le mode de sélection par tournoi ont été huit (8) à atteindre l'optimum global à égalité avec les exécutions avec un croisement non-élitiste et une sélection par rang.

Bilan de la convergence

D'après les deux séries d'expérimentations faites pour étudier la convergence de l'algorithme génétique, nous avons pu déduire que :

1. l'écart-type ne renseigne en rien si une exécution de l'algorithme génétique a atteint

Numéro de l'expérimentation	Méthode de croisement (normale ou élitiste)	Mode de sélection (rang ou tournoi)	Meilleure évaluation (E_b)	Première génération de la meilleure évaluation (G_b)	Nombre de générations avec la meilleure évaluation	Écart-type à la génération G_b	Génération du premier écart-type inférieur à $1E-12$	Nombre de générations avec un écart-type inférieur à $1E-12$	Nombre de générations ayant un écart-type inférieur à $1E-12$ et présentant la meilleure évaluation	L'algorithme finit-il avec la meilleure évaluation rencontrée lors de l'expérimentation?
21	normale	tournoi	0,9613	2	48	1,7940	30	14	14	oui
22	normale	tournoi	0,9613	3	47	2,0624	13	23	23	oui
23	normale	tournoi	0,9613	2	48	1,7414	20	14	14	oui
24	normale	tournoi	0,9513	11	39	1,4954	31	12	12	oui
25	normale	tournoi	0,9613	4	46	2,5264	13	21	21	oui
26	normale	tournoi	0,8921	17	33	2,0359	29	9	9	oui
27	normale	tournoi	0,9413	11	39	3,1055	41	5	5	oui
28	normale	tournoi	0,9613	5	45	3,6868	14	19	19	oui
29	normale	tournoi	0,9613	5	45	2.4786	27	17	17	oui
30	normale	tournoi	0,9513	13	37	1,5461	24	13	13	oui
31	élitiste	tournoi	0,9613	6	44	3,5702	N/A	0	0	oui
32	élitiste	tournoi	0,9613	6	44	4,2116	N/A	0	0	oui
33	élitiste	tournoi	0,9513	13	37	2,5744	N/A	0	0	oui
34	élitiste	tournoi	0,9613	3	47	1,9588	14	3	3	oui
35	élitiste	tournoi	0,9613	3	47	2,3519	N/A	0	0	oui
36	élitiste	tournoi	0,9313	12	38	1,9072	30	9	9	oui
37	élitiste	tournoi	0,9613	7	43	3,3669	14	19	19	oui
38	élitiste	tournoi	0,9613	10	40	2,2660	24	15	15	oui
39	élitiste	tournoi	0,9613	8	42	2,4776	15	21	21	oui
40	élitiste	tournoi	0,9613	4	46	2,6017	21	17	17	oui

TAB. III.7 – *Résultats des 20 dernières expérimentations sur la convergence de l'algorithme génétique*

ou non l'optimum global,

2. la taille de la population joue un rôle important dans la découverte de l'optimum global dans un cas comme celui-ci,

3. le nombre de générations ne semble pas jouer un grand rôle puisque dans la seconde expérimentation les exécutions ayant trouvé l'optimum global ont passé en moyenne 44,59 générations avec l'optimum comme meilleur individu.

Afin que le lecteur puisse se faire une idée sur l'adaptation possible, la figure III.5 montre le résultat d'une adaptation pour le document Web et le contexte de cette expérimentation.

FIG. III.5 – *Aperçu de l'affichage d'un document* ***adapté*** *de type Wiki sur un téléviseur.*

Enfin, il est nécessaire de s'intéresser au temps d'exécution de l'algorithme génétique lors de l'adaptation d'un document. Les figures III.6 et III.7 montrent les temps mis par l'algorithme génétique à chaque exécution respectivement de la première et seconde expérimentation.

Le temps d'exécution de l'algorithme génétique pour la première expérimention est compris entre 701 et 2583 secondes, avec une moyenne de 1280 secondes. Pour la seconde expérimentation, l'exécution de l'algorithme génétique prend entre 1291 et 2033 secondes.

L'agrandissement de la population entre les deux expérimentations a fait passer l'écart-type entre les temps d'exécution de 556 secondes à 177 secondes. Or le nombre d'évaluations est le même entre les deux expérimentations. On peut en conclure que les exécutions sont plus homogènes en terme de temps d'exécution avec une population plus importante.

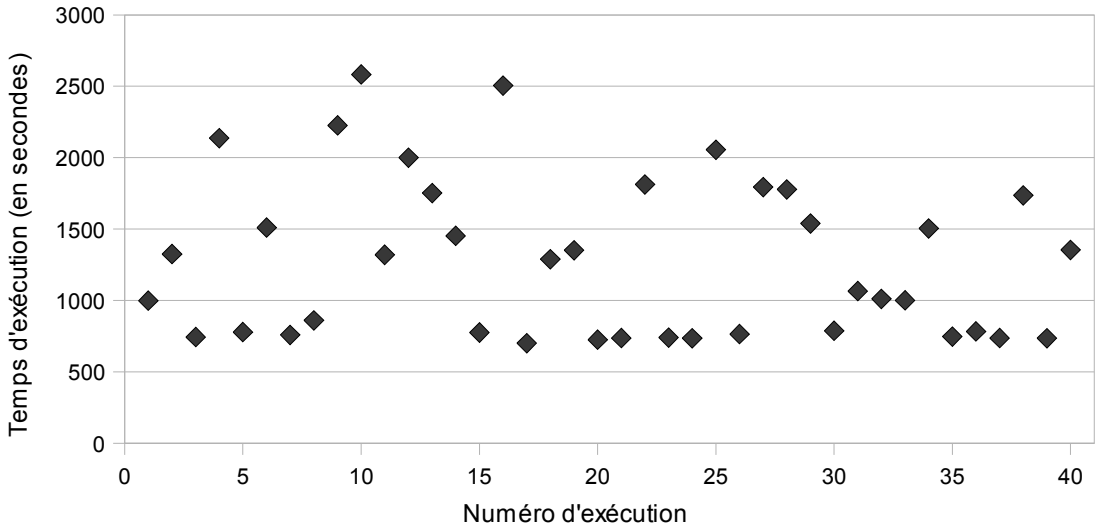

FIG. III.6 – *Temps d'exécution de l'algorithme génétique pour les exécutions de la première expérimentation.*

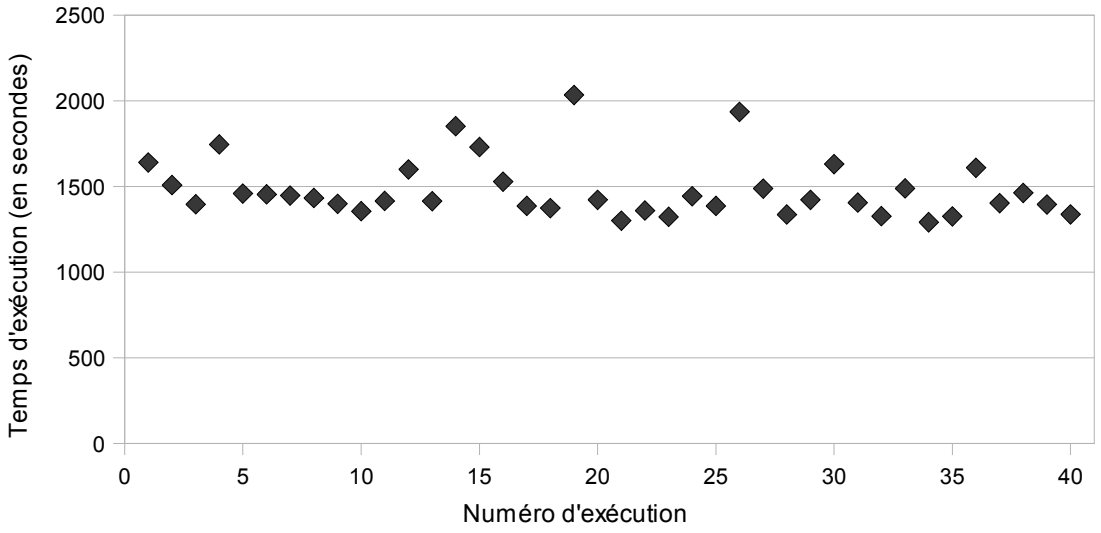

FIG. III.7 – *Temps d'exécution de l'algorithme génétique pour les exécutions de la seconde expérimentation.*

CHAPITRE III

Implémentations tests et évaluations

Conclusion

Ce chapitre a fait état de la mise en œuvre d'une plate-forme de tests et de l'analyse que nous pouvions faire à la lecture des résultats obtenus.
Nous avons en effet décrit la mise en place d'un banc d'essai complet proposant l'implémentation complète de notre modèle. Ces développements logiciels ont été entièrement réalisés dans le cadre de la thèse, et se basent sur le langage de programmation Java. Particulièrement nous avons programmé un à un les différents modules présentés dans le chapitre II.4.3, ainsi que l'algorithme génétique lui-même, les estimateurs et les transformations associées. Ce travail a été réalisé sur un serveur virtualisé afin de pouvoir évaluer la performance de notre système dans un environnement contrôlé.

Par la suite, nous avons conduit de nombreuses expériences sur ce système afin d'éprouver sa capacité à adapter des pages en un temps fini, et à converger vers un optimum. Nous avons notamment fait varier la méthode de croisement (normal ou élitiste), le mode de sélection des individus (par rang ou par tournoi), la taille de la population, et le nombre de générations de l'algorithme génétique. Nous en sommes arrivés à la conclusion empirique que l'algorithme converge vers un optimum.

Conclusion

Ce travail avait pour objectif d'adresser un des enjeux majeurs de l'informatique pervasive, à savoir l'adaptation automatique des contenus Web au dispositif de consultation. Le guichet d'entrée vers le Web autrefois limité à l'ordinateur de bureau équipé d'un clavier et d'une souris dans une salle dédiée, est aujourd'hui devenu plus répandu dans notre vie quotidienne (smartphone, interactive IPTV, tablet, etc.). Cela a fortement contribué à l'émergence de très nombreux dispositifs hétérogènes, et donc la difficulté d'adresser des mécanismes d'hypermédia adaptatifs à l'aide d'heuristiques.

Après avoir exposé l'état de l'art en classifiant et analysant les contributions que nous avons identifié dans la littérature (Lardon *et al.*, 2008), nous avons identifié des objectifs pour notre travail de thèse permettant de contribuer à ce domaine. Nous nous sommes alors engagés à proposer un modèle et de le vérifier empiriquement permettant la généricité de l'approche (s'applique à n'importe quelle cible [12]), l'universalité (cible non connue par avance), ainsi que la conduite de ces objectifs et de l'adaptation afférente en un temps raisonnable.

Les apports de cette thèse sont nombreux et résident sur plusieurs plans. Ils participent tous à l'objectif d'adapter un arbre DOM sans intervention de l'humain pour une cible non connue à l'avance, ce que nous qualifions d'hypermédia adaptatif.

Tout d'abord, nous avons démontré que rechercher la meilleure adaptation d'un document Web pour une cible donnée constitue un espace de recherche trop grand pour être exploré raisonnablement par un algorithme conventionnel. Nous avons donc établi que les méthodes à base d'heuristiques demeurent des méthodes *ad hoc* et ne peuvent adresser l'objectif d'universalité que nous nous sommes fixés dans ce travail.

Du point de vue modèle, nous avons alors choisi de considérer ce problème comme un problème d'approximation, et l'avons traité à l'aide de méta-heuristique. C'est alors que nous avons proposé un modèle complet intégrant un algorithme génétique comme moteur de l'adaptation, et entouré des modules nécessaires à la poursuite de la navigation (Lardon *et al.*, 2010a).

12. Nous avons appelé cible un dispositif permettant de consulter un document Web.

Par exemple, le mécanisme d'adaptation pour des cibles considérées comme "pauvres" requiert souvent le marcottage de l'arbre DOM, et donc "découper" des documents Web en plusieurs parties. Il faut alors suivre la navigation de l'utilisateur dans les sous pages nouvellement crées. La réalisation de ce modèle nous a également conduit à proposer un mécanisme original pour se faire, que nous avons nommé cookie transdomaine (Lardon *et al.*, 2010b).

La fonction d'évaluation est un élément clé des algorithmes génétiques. Nous avons développé une théorie basée sur les graphes pour évaluer au mieux les impacts des combinaisons d'adaptations possibles, sans pour autant conduire ces adaptations pour tous les individus utilisés dans notre algorithme génétique. L'avantage de l'approche est d'estimer les conséquences d'une adaptation sans la réaliser (via les concepts d'estimation et d'estimateur introduits dans notre modèle), afin d'optimiser l'exécution temporelle de l'algorithme).

Enfin, nous avons fait état de la mise en œuvre d'une plate-forme de tests et de l'analyse que nous pouvions faire à la lecture des résultats obtenus. Nous avons en effet décrit la mise en place d'un banc d'essai complet proposant l'implémentation de notre modèle. Ces développements logiciels ont été entièrement réalisés dans le cadre de la thèse, et se basent sur le langage de programmation Java. Ce travail a été réalisé sur un serveur virtualisé afin de pouvoir évaluer la performance de notre système dans un environnement contrôlé. L'algorithme converge à l'implémentation et donne des résultats exploitables en terme de navigation par l'humain.

Nous pouvons donc conclure que nous avons atteint nos objectifs de généricité et d'universalité. Une expérience utilisateur à très court terme viendra qualifier la pertinence de l'adaptation pour l'humain (critère d'utilisabilité), mais les premiers résultats démontrent que les adaptations produites par notre algorithme sont fonctionnelles (critère d'utilité satisfait). Le troisième objectif était le temps d'adaptation. Celui-ci est partiellement atteint : nous obtenons un optimum en un temps raisonnable (quelques minutes), mais ce n'est pas immédiat, et donc inexploitable en temps réel. Cela n'exclut pas une utilisation en édition pour une cible donnée, en substitution aux approches de rétro-ingénierie et transcodage présentées au chapitre I.2.3. Cela constitue très certainement une piste d'amélioration possible. Nous pensons qu'à moyen terme nous pourrions étudier s'il est possible de classifier les pages par leur séquence d'adaptation. Cela pourrait permettre de créer un algorithme frontal à notre moteur d'adaptation, qui, par apprentissage, déterminerait des classes de documents Web et ainsi les séquences d'adaptation qui sont

pertinentes en fonction de leurs caractéristiques (nombre de mots, type de liens, etc.). Si un tel algorithme peut exister, son apprentissage pourrait être supervisé par notre algorithme d'adaptation, afin de disposer *in fine* d'une bijection entre un document Web non connu à l'avance, et l'adaptation à lui appliquer. Le traitement du choix de la séquence d'adaptation à conduire serait alors immédiat.

Annexes

ANNEXE A

Compléments d'analyse à l'étude bibliographique

Diagrammes de répartition

Par origine géographique Afin de replacer notre étude bibliographique dans un cadre géographique, nous avons reporté le nombre des publications retenues pour chaque pays du globe dans la figure A.1. Dans le cas d'une publication entre laboratoires de différents pays, le crédit a été donné à chacun des pays. Ainsi le compte des publications sur la répartition géographique est de 51, ce qui correspond aux 47 publications de notre état de l'art auquel s'ajoute le nombre de publications créditées pour deux pays, c'est-à-dire 4.

De cette répartition, il est possible de découper trois zones géographiques d'activité scientifique le plus soutenue sur le domaine, à savoir l'Amérique du Nord, l'Europe et l'Asie. En effet l'Amérique du Nord totalise 17 publications (18 publications moins une partagée entre les États-Unis et le Canada), l'Europe suit avec 18 publications (un total de 19, auquel il faut enlever 1 car une publication est le fruit d'une collaboration italo-belge) et l'Asie regroupe 8 publications. Les zones de moindre activité sur le domaine sont disséminées autour du globe : Le Brésil pour l'Amérique du Sud, l'Océanie avec l'Australie et Singapour, l'est du bassin méditéranéen avec Chypre et le Liban et l'Afrique avec l'Afrique du Sud. En résumé chaque continent a eu une activité scientifique dans le domaine depuis les dix dernières années.

Par année de publication Notre étude étant placée géographiquement grâce à la partie précédente, il est nécessaire de se pencher sur la répartition par année des publications retenues.

Avant d'analyser la figure figure A.2, il est important de rappeler au lecteur que les 47 publications retenues l'ont été parmi celles publiées depuis 1999 et que sur plusieurs articles traitant du même projet scientifique, seule la publication la plus récente a été conservée.

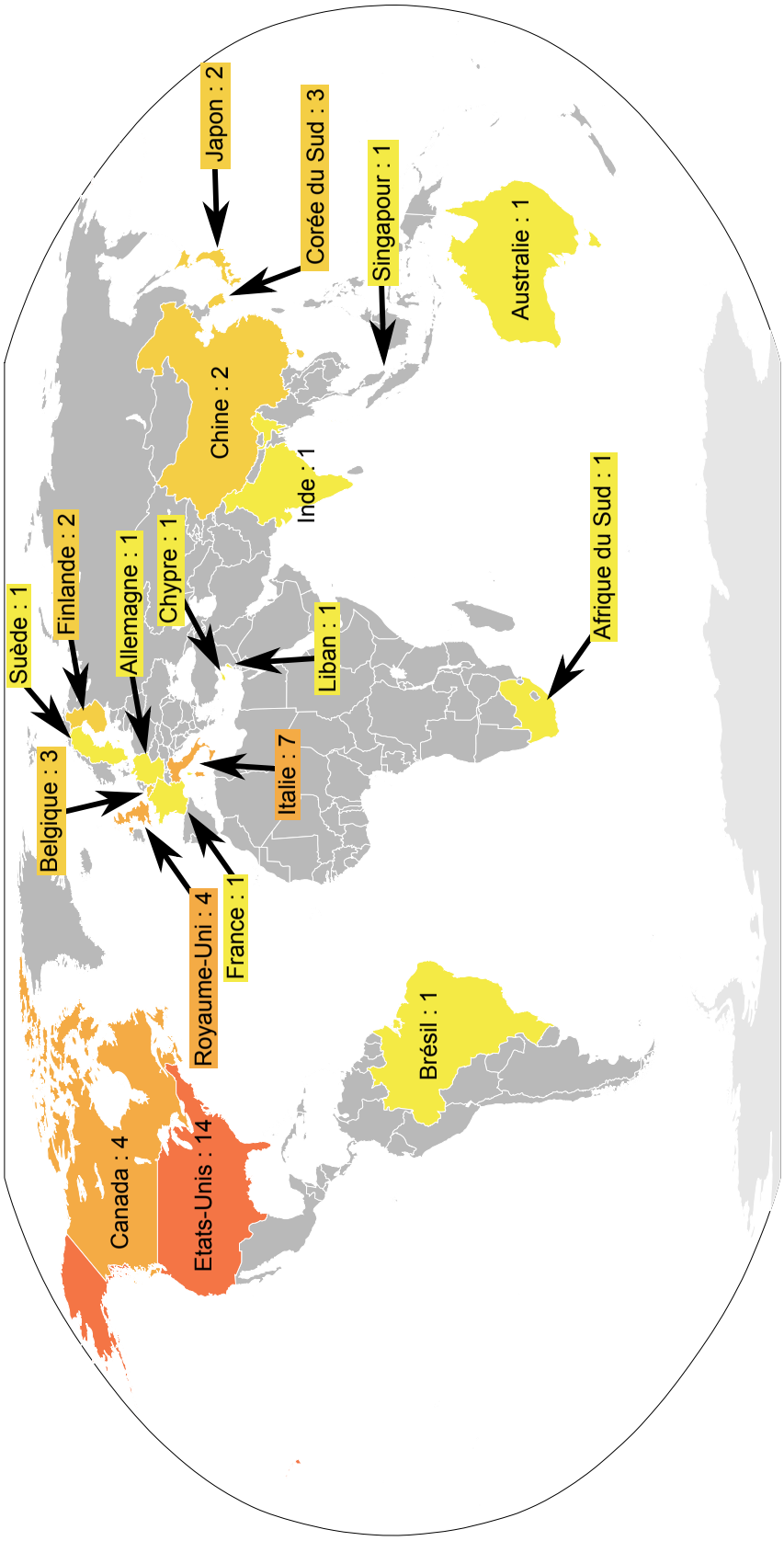

FIG. A.1 – *Cadre géographique de notre étude : répartition par pays des publications retenues.*

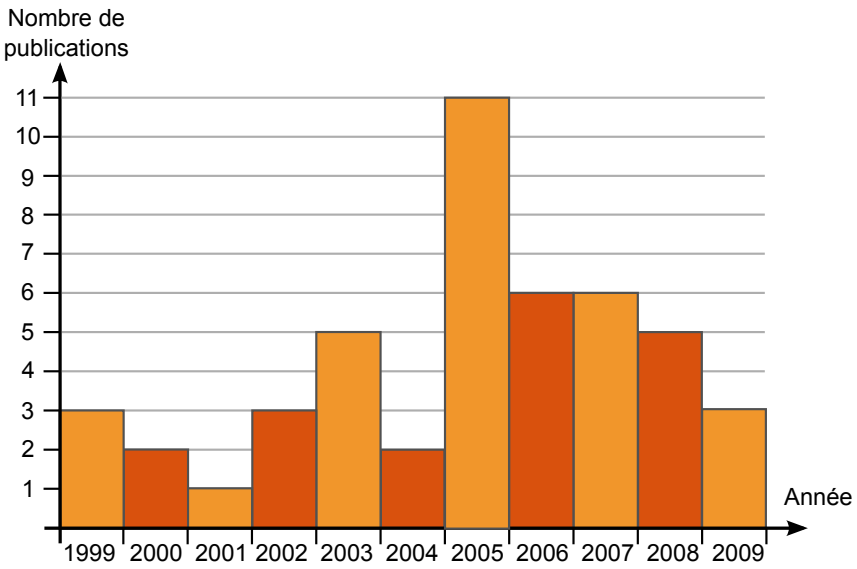

FIG. A.2 – *Répartition temporelle de notre étude : nombre d'articles par année de publication.*

Ces précisions étant rappellées, nous pouvons noter que les publications entre 1999 et 2004 représentent un peu plus d'un tiers du total (16 sur les 47). Les articles écrits entre les années 2005 et 2007 totalisent 23, c'est-à-dire près de la moitié de ceux retenus. Enfin les années 2008 et 2009 n'ont vu que 8 des publications retenues.

ANNEXE B

Exemple de document Web

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Il était une fois ...</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style>
body {
width: 640px;
font-size: 24px;
}

a {
margin-left: 20px;
}
</style>
</head>
<body>

Il était une fois ...
<a href="suite.html">
<img src="portail.png" alt="porte vers la suite du récit" />
</a>

</body>
</html>
```

ANNEXE C

Exemple de fichier de configuration

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>Configuration du système d'adaptation</comment>
<!-- UTILISATION D'UN PROXY OU NON -->
<entry key="useProxy">yes</entry>
<!--<entry key="useProxy">no</entry>-->
<!-- ADRESSE DU PROXY -->
<entry key="proxyHost">cache.univ-st-etienne.fr</entry>
<!-- PORT DU PROXY -->
<entry key="proxyPort">3128</entry>
<!-- CHEMIN VERS LA BASE DES RÉPERTOIRES DE JOURNALISATION -->
<entry key="loggingPathBase">/home/dje/Desktop/testDTEC/</entry>

<!-- CONFIGURATION DU MODULE DE SESSION -->

<!-- JOURNALISATION DES ESTIMATIONS AVANT ET
APRÈS LES EXÉCUTIONS DU MOTEUR DE TRANSFORMATION -->
<entry key="estimationLogging">yes</entry>

<!-- ??? TRANSFORMATIONS UTILISÉES ??? -->

<!-- MALUS À LA PERFORMANCE DE LA TRANSFORMATION Layering -->
<entry key="layeringPerfMalus">3</entry>
<!-- NOMBRE MAXIMUM DE NOEUDS SUR LESQUELS
LA TRANSFORMATION Layering S'APPLIQUE -->
<entry key="maxLayering">5</entry>
<!-- CHOIX POUR L'AFFICHAGE DE SOUS-PAGES : un lien/image suite ou non -->
```

```
<entry key="layeringLinkNavigation">true</entry>
<!-- MALUS À LA PERFORMANCE DE LA TRANSFORMATION FontScaling -->
<entry key="fontScalingPerfMalus">1</entry>
<!-- MALUS À LA PERFORMANCE DE LA TRANSFORMATION Text2Voice -->
<entry key="text2VoicePerfMalus">2</entry>
<!-- MALUS À LA PERFORMANCE DE LA TRANSFORMATION Wiki2VoiceAndLinks -->
<entry key="wiki2VoiceAndLinksPerfMalus">2</entry>
<!-- CHOIX DE SAVOIR SI ON VOCALIZE OU SEULEMENT ÉCRIRE LES PAROLES -->
<entry key="vocalize">no</entry>
<!-- CHOIX DE LA VOIX SI ON VOCALIZE -->
<entry key="vocalizeVoice">3</entry>

<!-- JOURNALISATION DANS LE MODULE DE SESSION -->
<!-- 0 pour aucune journalisation -->
<!-- 1 pour une journalisation dans le fichier session.log -->
<!-- 2 pour la journalisation en console -->
<entry key="sessionLogging">2</entry>

<!-- CONFIGURATION DU MOTEUR DE TYRANSFORMATION -->

<!-- NOMBRE D'INDIVIDUS DE LA POPULATION DE L'ALGORITHME GÉNÉTIQUE -->
<entry key="populationSize">1000</entry>
<!-- NOMBRE DE GÉNÉRATIONS AVANT LA FIN DE L'ALGORITHME GÉNÉTIQUE -->
<entry key="maxGeneration">50</entry>
<!-- CONFIGURATION DU CROISEMENT : elististe ou non -->
<entry key="UseElitistMethod">true</entry>
<!-- TAUX DE CROISEMENT DE L'ALGORITHME GÉNÉTIQUE -->
<entry key="crossoverRate">0.6</entry>
<!-- TAUX DE MUTATION DE L'ALGORITHME GÉNÉTIQUE -->
<entry key="mutationRate">0.001</entry>
<!-- CONFIGURATION DE LA SELECTION :
par roue biaisée sur le rang (true) ou par tournoi (false) -->
<entry key="UseRankingSelection">false</entry>
<!-- JOURNALISATION DANS L'ALGORITHME GÉNÉTIQUE -->
<!-- 0 pour aucune journalisation -->
<!-- 1 pour une journalisation des estimations-types :
meilleure, pire, moyenne, médiane -->
<!-- 2 pour une journalisation des estimations-types :
meilleure, pire, moyenne, médiane,
```

```
et une journalisation des individus-type :
meilleur, pire, médian -->
<!-- 3 pour comme précédent +
création des pages correspondantes aux meilleurs -->
<!-- 4 pour comme précédent +
création des pages correspondantes aux pires individus
de chaque génération -->
<entry key="AGLogging">2</entry>
<!-- PARALLÈLISATION DES ÉVALUATIONS DES INDIVIDUS
DANS L'ALGORITHME GÉNÉTIQUE -->
<entry key="evaluationThreadNumber">2</entry>
<!-- POIDS DE LA LARGEUR ESTIMÉE DE LA PAGE -->
<entry key="widthCoef">0.4</entry>
<!-- POIDS DE LA PROFONDEUR DE L'ARBRE DOM -->
<entry key="depthCoef">0.15</entry>
<!-- POIDS DU NOMBRE ESTIMÉ DE MOTS -->
<entry key="wordCoef">0.4</entry>
<!-- POIDS DE LA PERFORMANCE -->
<entry key="perfCoef">0.05</entry>
</properties>
```

Bibliographie

Abowd Gregory D., Dey Anind K., Brown Peter J., Davies Nigel, Smith Mark & Steggles Pete. **1999**. Towards a Better Understanding of Context and Context-Awareness. *Pages 304–307 de : HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag.

Adipat Boonlit & Zhang Dongsong. **2005**. Adaptive and Personalized Interfaces for Mobile Web. *SSRN eLibrary*.

Arase Yuki, Maekawa Takuya, Hara Takahiro, Uemukai Toshiaki & Nishio Shojiro. **2007**. A web browsing system for cellular-phone users based on adaptive presentation. *Univers. Access Inf. Soc.*, **6**(3), 259–271.

Ardon S., Gunningberg P., Landfeldt B., Ismailov Y., Portmann M. & Seneviratne A. **2003**. MARCH: A distributed content adaptation architecture. *International Journal of Communication Systems*, **16**, 97–115.

Artail A. & Raydan Mackram. **2005**. Device-aware desktop web page transformation for rendering on handhelds. *Personal Ubiquitous Comput.*, **9**(6), 368–380.

Baluja Shumeet. **2006**. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. *Pages 33–42 de : WWW '06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA : ACM.

Berhe Girma, Brunie Lionel & Pierson Jean-Marc. **2005**. Distributed Content Adaptation for Pervasive Systems. *Information Technology: Coding and Computing, International Conference on*, **2**, 234–241.

Berti Silvia & Paternò Fabio. **2005**. Migratory MultiModal interfaces in MultiDevice environments. *Pages 92–99 de : ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*. New York, NY, USA : ACM.

Bickmore Timothy W., Girgensohn Andreas & Sullivan Joseph W. **1999**. Web Page Filtering and Re-Authoring for Mobile Users. *Computer Journal*, **42**(6), 534–46.

Björk Staffan, Holmquist Lars Erik, Redström Johan, Bretan Ivan, Danielsson Rolf, Karlgren Jussi & Franzén Kristofer. **1999**. WEST: a Web browser for small terminals. *Pages 187–196 de : UIST '99: Proceedings of the 12th*

annual ACM symposium on User interface software and technology. New York, NY, USA : ACM.

Borodin Yevgen, Mahmud Jalal & Ramakrishnan I.V. **2007**. Context browsing with mobiles - when less is more. *Pages 3–15 de : MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*. New York, NY, USA : ACM.

Bouillon Laurent, Limbourg Quentin, Vanderdonckt Jean & Michotte Benjamin. **2005**. Reverse Engineering of Web Pages based on Derivations and Transformations. *Dans : Web Congress, 2005. LA-WEB 2005. Third Latin American*.

Buchholz Sven, Schill Alexander & Schill Er. **2003**. Adaptation-Aware Web Caching: Caching in the Future Pervasive Web. *Pages 26–28 de : In 13th GI/ITG Conference Kommunikation in verteilten Systemen (KiVS*.

Burigat Stefano, Chittaro Luca & Gabrielli Silvia. **2008**. Navigation techniques for small-screen devices: An evaluation on maps and web pages. *Int. J. Hum.-Comput. Stud.*, **66**(2), 78–97.

Buyukkokten Orkut, Kaljuvee Oliver, Garcia-Molina Hector, Paepcke Andreas & Winograd Terry. **2002**. Efficient web browsing on handheld devices using page and form summarization. *ACM Trans. Inf. Syst.*, **20**(1), 82–115.

Casteleyn Sven, Plessers Peter & Troyer Olga De. **2006**. On Generating Content and Structural Annotated Websites Using Conceptual Modeling. *Conceptual Modeling - ER 2006*, **4215/2006**, 267–280.

Cena Federica, Console Luca, Gena Cristina, Goy Anna, Levi Guido, Modeo Sonia & Torre Ilaria. **2006**. Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Commun.*, **19**(4), 369–384.

de Oliveira Rodrigo & da Rocha Heloísa Vieira. **2007**. Consistency on Multi-device Design. *Human-Computer Interaction – INTERACT 2007*, **4663/2009**, 617–623.

De Virgilio Roberto, Torlone Riccardo & Houben Geert-Jan. **2007**. Rule-based Adaptation of Web Information Systems. *World Wide Web*, **10**(4), 443–470.

Di Lucca Giuseppe Antonio, Fasolino Anna Rita & Tramontana Porfirio. **2005**. Recovering Interaction Design Patterns in Web Applications. *Pages 366–374 de : CSMR '05: Proceedings of the Ninth European Conference on Software Maintenance and Reengineering*. Washington, DC, USA : IEEE Computer Society.

Ferretti Stefano, Roccetti Marco & Palazzi Claudio E. **2007**. Web Content Search and Adaptation for IDTV: One Step Forward in the Mediamorphosis Process toward Personal-TV. *Advances in Multimedia*, **2007**.

Fox Armando, Gribble Steven D., Chawathe Yatin & Brewer Eric A. **1999**. Adapting to network and client variation using infrastructured proxies: lessons and perspectives. *Pages 431–446 de : Mobility: processes, computers, and agents*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co.

Gaeremynck Yves, Bergman Lawrence D. & Lau Tessa. **2003**. MORE for Less: Model Recovery from Visual Interfaces for Multi-Device Application Design. *Pages 69–76 de : IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*. New York, NY, USA : ACM.

Gerber Anna, Lawley Michael, Raymond Kerry, Steel Jim & Wood Andrew. **2002**. Transformation: The Missing Link of MDA. *Pages 90–105 de : ICGT '02: Proceedings of the First International Conference on Graph Transformation*. London, UK : Springer-Verlag.

Goldberg David E. **1989**. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc.

Goldberg David E. & Deb Kalyanmoy. **1991**. A comparative analysis of selection schemes used in genetic algorithms. *Pages 69–93 de : Foundations of Genetic Algorithms*.

Grassel Guido, Geisler Roland, Vartiainen Elina, Chauhan Deepika & Popescu Andrei. **2006**. The Nokia Open Source Browser. *Dans : Proceedings of MobEA IV*.

Gupta Suhit, Becker Hila, Kaiser Gail & Stolfo Salvatore. **2006**. Verifying genre-based clustering approach to content extraction. *Pages 875–876 de : WWW '06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA : ACM.

Holland John H. **1992**. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA : MIT Press.

Hori Masahiro, Kondoh Goh, Ono Kouichi, Hirose Shin-ichi & Singhal Sandeep. **2000**. Annotation-based Web content transcoding. *Pages 197–211 de : Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*. Amsterdam, The Netherlands, The Netherlands : North-Holland Publishing Co.

Hwang Yonghyun, Kim Jihong & Seo Eunkyong. **2003**. Structure-Aware Web Transcoding for Mobile Devices. *IEEE Internet Computing*, **7**(5), 14–21.

Hübsch Gerald, Springer Thomas, Spriestersbach Axel & Ziegert Thomas. **2005**. An Integrated Platform for Mobile, Context-Aware, and Adaptive Enterprise Applications. *Wirtschaftsinformatik 2005*, 1105–1124.

Jones Matt, Buchanan George & Mohd-Nasir Norliza. **1999**. An Evaluation of WebTwig - A Site Outliner for Handheld Web Access. *Pages 343–345 de : HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag.

Joshi Anupam. **2000**. On proxy agents, mobility, and web access. *Mob. Netw. Appl.*, **5**(4), 233–241.

jung Han Eun, onn Wong Chee, chul Jung Kee, ho Lee Kyung & yi Kim Eun. **2009**. Efficient page layout analysis on small devices. *Journal of Zhejiang University - Science A*, **Volume 10, Number 6 / June, 2009**, 800–804.

Komninos Andreas & Milligan Chris. **2007**. Increasing the speed of Information Access on the Mobile web using HTML feature extraction. *Dans : 30th Annual International ACM SIGIR Conference, Web Information-Seeking and Interaction Workshop*.

Korva Jari, Plomp Johan, Määttä Petri & Metso Maija. **2001**. On-Line Service Adaptation for Mobile and Fixed Terminal Devices. *Pages 252–259 de : MDM '01: Proceedings of the Second International Conference on Mobile Data Management*. London, UK : Springer-Verlag.

Lam Heidi & Baudisch Patrick. **2005**. Summary thumbnails: readable overviews for small screen web browsers. *Pages 681–690 de : CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM.

Lardon Jérémy, Ates Mikaël, Gravier Christophe & Fayolle Jacques. **2008**. Overview of Web Content Adaptation. *Pages 384–387 de : ICEIS 2008: Proceedings of the 10th International Conference on Enterprise Information Systems*.

Lardon Jérémy, Gravier Christophe & Fayolle Jacques. **2010a**. DOM tree estimation and computation: overview of a new web content adaptation system. *Pages 357–360 de : EICS '10: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*. New York, NY, USA : ACM.

Lardon Jérémy, Gravier Christophe & Fayolle Jacques. **2010b**. Towards a unified device characteristic retrieval and propagation : context proxy presentation. *Dans : ICME 2010: Proceedings of the 2010 IEEE International Conference on Multimedia and Expo*.

Liu Ling, Buttler David, Caverlee James, Pu Calton & Zhang Jianjun. **2005**. A methodical approach to extracting interesting objects from dynamic web pages. *Int. J. Web Grid Serv.*, **1**(2), 165–195.

Lunn Darren, Bechhofer Sean & Harper Simon. **2008**. A user evaluation of the SADIe transcoder. *Pages 137–144 de : Assets '08: Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*. New York, NY, USA : ACM.

MacKay Bonnie, Watters Carolyn & Duffy Jack. **2004**. Web Page Transformation When Switching Devices. *Mobile Human-Computer Interaction – MobileHCI 2004*, **3160/2004**, 228–239.

Maheshwari Anuj, Sharma Aashish, Ramamritham Krithi & Shenoy Prashant. **2002**. TranSquid: Transcoding and Caching Proxy for Heterogenous E-Commerce Environments. *Page 50 de : RIDE '02: Proceedings of the 12th International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems (RIDE'02)*. Washington, DC, USA : IEEE Computer Society.

Mahmud Jalal, Borodin Yevgen, Ramakrishnan I. V. & Ramakrishnan C. R. **2009**. Automated construction of web accessibility models from transaction clickstreams. *Pages 871–880 de : WWW '09: Proceedings of the 18th international conference on World wide web*. New York, NY, USA : ACM.

Milic-Frayling Natasa, Sommerer Ralph, Rodden Kerry & Blackwell Alan. **2004**. SmartView and SearchMobil: Providing Overview and Detail in Handheld Browsing. *Mobile and Ubiquitous Information Access*, **2954/2004**, 17–19.

Mohomed Iqbal, Cai Jim Chengming, Chavoshi Sina & de Lara Eyal. **2006**. Context-aware interactive content adaptation. *Pages 42–55 de : MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*. New York, NY, USA : ACM.

Moshchuk Alex, Gribble Steven D. & Levy Henry M. **2008**. Flashproxy: transparently enabling rich web content via remote execution. *Pages 81–93 de : MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA : ACM.

Médiamétrie. **2009a** (Août). *La fréquentation des sites Internet français en Août 2009*. Communiqué de presse.

Médiamétrie. **2009b** (Novembre). *L'Internet aujourd'hui et demain*. Communiqué de presse.

Otterbacher Jahna, Radev Dragomir & Kareem Omer. **2008**. Hierarchical summarization for delivering information to mobile devices. *Inf. Process. Manage.*, **44**(2), 931–947.

Paganelli Laila & Paternò Fabio. **2003**. A Tool for Creating Design Models from Web Site Code. *International Journal of Software Engineering and Knowledge Engineering*, **13**(2), 169–189.

Song Kisub & Lee Kyong-Ho. **2008**. Generating multimodal user interfaces for Web services. *Interact. Comput.*, **20**(4-5), 480–490.

Thevenin David. **2001**. *Adaptation en Interaction Homme-Machine: Le cas de la Plasticité*. Thèse de doctorat, Université Joseph Fourier, Grenoble.

Watters Carolyn & Zhang Rui. **2003**. PDA Access to Internet Content: Focus on Forms. *Page 105.1 de : HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 4*. Washington, DC, USA : IEEE Computer Society.

West Philip. **2005**. *A Framework for Responsive Content Adaptation in Electronic Display Networks*. M.Phil. thesis, Rhodes University, South Africa.

Wobbrock Jacob O., Forlizzi Jodi, Hudson Scott E. & Myers Brad A. **2002**. WebThumb: interaction techniques for small-screen browsers. *Pages 205–208 de : UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*. New York, NY, USA : ACM.

Xiao Xiangye, Luo Qiong, Hong Dan, Fu Hongbo, Xie Xing & Ma Wei-Ying. **2009**. Browsing on small displays by transforming Web pages into hierarchically structured subpages. *ACM Trans. Web*, **3**(1), 1–36.

Xie Xing, Miao Gengxin, Song Ruihua, Wen Ji-Rong & Ma Wei-Ying. **2005**. Efficient Browsing of Web Search Results on Mobile Devices Based on Block Importance Model. *Pages 17–26 de : PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA : IEEE Computer Society.

Yin Xinyi & Lee Wee Sun. **2005**. Understanding the function of web elements for mobile content delivery using random walk models. *Pages 1150–1151 de : WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*. New York, NY, USA : ACM.

61 références.