# Knowledge-based configuration : a contribution to generic modeling, evaluation and evolutionary optimization

Luis Garcés Monge

# THÈSE

En vue de l'obtention du

**DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE**

**Délivré par :**

IMT - École Nationale Supérieure des Mines d'Albi-Carmaux

**Présentée et soutenue par :**

**Luis GARCÉS MONGE**

**le** 11 octobre 2019

**Titre :**

Knowledge-based configuration: a contribution to generic modeling, evaluation and evolutionary optimization

**École doctorale et discipline ou spécialité :**
EDSYS : Génie Industriel 4200046

**Unité de recherche :**
Centre Génie Industriel, IMT Mines Albi

**Directeur/trice(s) de Thèse :**
Michel ALDANONDO, Professeur, IMT Mines Albi
Paul PITIOT, Enseignant-chercheur, 3IL Rodez

**Jury :**

Michel TOLLENAERE, Professeur, Grenoble INP, Président
Pierre Alain YVARS, Professeur, Inst. Supérieur de Mécanique de Paris, Rapporteur
Éric BONJOUR, Professeur, Université de Lorraine, Rapporteur
Catherine DA CUNHA, Maître de Conférences, Centrale Nantes, Examinateur
Élise VAREILLES, Maître-assistant, IMT Mines Albi, Examinateur

# Acknowledgements

To God for giving me patience and perseverance to end this stage of my life.
To my parents José Luis Garcés Fernández and Vera Virginia Monge Mora for all the love and support they give me and for always giving me the best.

To Michel Aldanondo, Paul Pitiot and Elise Vareilles for all the dedication and patience towards me. My eternal thanks
Especially to Michel Aldanondo for his advice, experience and knowledge and to Paul Pitiot and Elise Vareilles for their knowledge, enthusiasm, guidance and professional support.
To my girlfriend Ariana Araya Madriz for all the support and compression.
To my friend Rafael Torres for all the support in the most difficult moments.
To my friend Marcela Meneses for the moral support and for encouraging me to undertake this professional adventure.
To my friend and partner Abdourahim Sylla for helping me during my stays in France.
To professor Marcos Moya for all the help and professional advice.
To professor Eldon Caldwell for your friendship and support in this process.
To my friend Isabelle Fournier (R.I.P), for the french language and culture classes. I know you must be very happy.
To all the colleagues of the Industrial Engineering Laboratory of IMT Mines Albi for all the support and good memories.
To the Instituto Tecnológico de Costa Rica and especially to the colleagues of the School of Production Engineering for all the support and for giving me the opportunity to live this experience.
To the members of the jury M. Michel Tollenaere, M. Eric Bonjour, M. Pierre Alain Yvars and Mme Catherine Da Cunha for all the recommendations and for having accepted to be part of my jury.

# Contents

## List of Figures

## List of Tables

# Résumé long de la thèse

# Configuration à base de connaissances : une contribution à la modélisation générique, à l'évaluation et à l'optimisation évolutionnaire

## 1. Introduction

Dans un contexte de personnalisation de masse, la configuration concourante du produit et de son processus d'obtention constitue un défi industriel important : de nombreuses options ou alternatives, tant sur les aspects Produit que Processus d'obtention, de nombreux liens ou contraintes et un besoin d'optimisation des choix réalisés doivent être pris en compte. Ce problème est intitulé O-CPPC (Optimization of Concurrent Product and Process Configuration). Nous considérons ce problème comme un CSP (Constraints Satisfaction Problem) et l'optimisons avec des algorithmes évolutionnaire. Un état de l'art (chapitre 2) fait apparaître : i) que la plupart des travaux de recherche sont illustrés sur des exemples spécifiques à un cas industriel ou académique et peu représentatifs de la diversité existante ; ii) un besoin d'amélioration des performances d'optimisation afin de gagner en interactivité et faire face à des problèmes de taille plus conséquentes. En réponse au premier point, ces travaux de thèse proposent les briques d'un modèle générique du problème O-CPPC (chapitre 3). Ces briques permettent d'architecturer le produit et son processus d'obtention, de décrire chaque sous-ensemble, composant et activité, de définir leurs contraintes, leur densité et les critères d'optimisation. Ce modèle générique est utilisé pour générer un benchmark réaliste pour évaluer les algorithmes d'optimisation. Ce benchmark est ensuite utilisé pour analyser la performance de l'approche évolutionnaire CFB-EA (chapitre 4). L'une des forces de cette approche est de proposer rapidement un front de Pareto proche de l'optimum. Pour répondre au second point, une amélioration de cette méthode est proposée puis évaluée (chapitre 5). L'idée est, à partir d'un premier front de Pareto approximatif déterminé très rapidement, de demander à l'utilisateur de choisir une zone d'intérêt (en fonction du prix et du délai d'obtention) et de restreindre la recherche de solutions uniquement sur cette zone. Cette amélioration entraine des gains de temps de calcul importants. Ce mémoire se termine par des perspectives de recherche pour chaque problématique de recherche abordée (chapitre 6).

# 2. Problème et problématiques de recherche associées

## 2.1 Le problème O-CPPC et les outils utilisés

Le problème O-CPPC est composé de quatre domaines : la configuration de produit, la configuration du processus d'obtention du produit, le couplage Produit/Projet et l'optimisation de cet ensemble au regard de différents objectifs.

La configuration de produit est définie comme la spécialisation d'un modèle de produit générique par rapport aux besoins spécifique d'un client donné. Deux points de vue peuvent être envisagés dans un modèle de configuration : le point de vue physique où le produit est décomposé en sous-ensembles et composants, ou le point de vue fonctionnel où le produit est décrit par un ensemble de fonctions de service. Le modèle générique proposé associe ces deux types de description dans un modèle dit « physico-fonctionnel ». L'activité de configuration correspond alors à trouver une instanciation du modèle de produit générique, c'est-à-dire une sélection de composants ou de niveaux fonctionnels, qui satisfont les besoins de l'utilisateur et les contraintes du produit.

La configuration du processus d'obtention est également définie comme la recherche d'une instanciation d'un modèle générique de processus d'obtention satisfaisant les besoins de l'utilisateur. Le modèle générique du processus décrit l'ensemble des opérations nécessaires pour obtenir un produit, leurs séquencements ainsi que le dimensionnement des ressources nécessaires. Il fait également apparaître des contraintes de compatibilités entre les ressources des différentes opérations.

L'unification de ces deux problèmes dans un modèle commun appelé configuration concourante Produit/Processus (CPPC) permet d'éviter les incohérences qui peuvent apparaître si l'on configure un aspect séparément de l'autre. Le modèle CPPC est défini comme l'union des modèles Produit/Processus additionné de contraintes de couplage. Il est modélisé grâce au formalisme des CSP par un ensemble de variables dont les valeurs sont définies dans un domaine et reliées par un ensemble de contraintes. Le modèle correspond à une représentation de haut niveau du produit et du processus associé. Nous assumons que les variables de décision sont discrètes (numérique ou symbolique) et que les contraintes les reliant sont donc des tables de compatibilités.

Ce modèle fait apparaître de nombreuses variables de décision. Les besoins du client portent sur un part limitée de ces variables. La saisie de ces besoins se fait lors d'une étape de configuration interactive avec le client. Une fois les besoins saisis, il reste, la plupart du temps, de nombreux choix dans le modèle (i.e. des variables non instanciées). Une étape d'optimisation va alors rechercher parmi les configurations possibles celles qui satisfont les objectifs du client (ici seul le coût et le temps de cycle sont considérés). En conséquence, le modèle proposé, illustré sur la figure 3, fait apparaître des variables continues représentant les coûts ou les durées des opérations ; ainsi que des contraintes reliant ces variables dites d'évaluation et les variables de décision.

Le modèle résultant appelé O-CPPC (Optimization of Concurrent Product-Processus Configuration) correspond donc au quadruplet <V, D, C, f> où V est l'ensemble des variables, D l'ensemble des domaines des variables de V, C l'ensemble des contraintes entre ces variables et f la fonction d'optimisation multi-objectif. La démarche d'aide à la décision proposée correspond à une résolution *a posteriori* de l'antagonisme entre les objectifs. L'optimisation doit donc fournir, en un temps raisonnable, un ensemble de solutions optimisées (front de Pareto). La démarche correspondante est illustrée sur la figure 2. La notion de temps de calcul raisonnable dépend du système Produit/Processus optimisé. Dans une situation de B2B avec un coût de produit élevé et un grand nombre de variables, le temps de calcul admissible peut atteindre une journée.

Le problème d'optimisation envisagé se caractérise par ses aspects combinatoire (variables de décision discrètes), multi-objectif (deux objectifs dans ces travaux), son espace de recherche fluctuant (selon le nombre de variables de décisions non instanciées à l'issue de la première phase de configuration interactive) et important (nous envisageons jusqu'à $10^{80}$ solutions potentielles sans prendre en compte les contraintes) et la présence de contraintes. Dans des travaux précédents, l'équipe de recherche a proposé une première évolution des algorithmes évolutionnaires appelée CFB-EA (Constraints Filtering Based Evolutionary Algorithm) comme méthode d'optimisation. L'algorithme de la méthode CFB-EA, illustré sur la figure 8, met en jeu une version de la méthode SPEA2 adaptée par l'inclusion de contraintes. Il intègre un moteur de filtrage de CSP permettant de préserver la faisabilité (cohérence par rapport aux contraintes) des individus générés à chaque étape de l'algorithme évolutionnaire (génération, croisement, mutation).

## 2.2 Problématiques de recherche associées

De nombreuses recherches étudient le problème de configuration de produit et/ou de processus. La plupart se concentre sur un seul aspect sans prendre en compte la relation étroite entre le produit et son processus d'obtention. Quelques études proposent un modèle concourant Produit/Processus et, à notre connaissance, aucune n'y associe l'optimisation du modèle correspondant. Parmi les travaux concernant l'optimisation de configuration de Produit, la plupart sont soit des cas académiques théoriques, soit des cas particuliers à une application donnée. Aucune étude ne met en place un véritable plan d'expérience, prenant en compte une grande diversité de situations industrielles, permettant d'analyser et d'évaluer l'étape d'optimisation (choix et paramétrage d'une méthode d'optimisation).

En conséquence, trois problématiques de recherche (Questions de Recherche) sont définies :

QR1 : Est-il possible de définir un modèle générique du problème O-CPPC capable de représenter une grande variété de cas réels et d'évaluer ainsi correctement les méthodes d'optimisation ?

QR2 : Quel est l'impact des caractéristiques-clés du problème O-CPPC (taille, niveau de contraintes, etc.) sur l'algorithme d'optimisation CFB-EA ?

QR3 : Est-il possible d'améliorer le temps d'exécution des algorithmes d'optimisation, spécialement du CFB-EA ?

Le chapitre 2 de la thèse établit un positionnement des travaux par rapport à un état de l'art des approches de modélisation (CSP) et d'optimisation (EA) utilisées. Puis les chapitres 3, 4, et 5 répondent respectivement à chacune des problématiques de recherche annoncées.

## 3. Modèle générique du problème O-CPPC

### 3.1 Modèle générique de configuration de Produit

De nombreuses études proposent de caractériser la conception de produit et/ou processus sous forme de décompositions hiérarchique selon différentes vues : fonctionnelle, besoins, composants physiques, processus, etc. Une vue de haut niveau de la décomposition physique d'un produit configurable met en jeu un certain nombre de composants regroupés en famille de composants (foc, family of components). Le produit et ses composants peuvent également être caractérisés par un ensemble d'attributs fonctionnels descriptifs (fdv, functional descriptive variables). Nous proposons de définir la configuration de produit comme l'instanciation des variables de décision. Ces variables de décision sont reliées par des contraintes de configuration qui limitent les combinaisons possibles de leurs valeurs.

Un autre aspect commun aux études existantes correspond à la modularité du produit configurable. Le produit peut être décomposé en différents modules associés aux fonctions ou composants principaux. Le modèle générique de produit configurable correspond donc à un ensemble de modules dits physico-fonctionnels. Chaque module est décrit par un ensemble de variables de décision (foc et fdv). Par exemple, pour la configuration d'une voiture, les modules pourraient être : la motorisation, le système électrique, la transmission, etc. Parmi les contraintes de configuration, on pourra alors distinguer les contraintes entre les variables d'un même module (intra-module) et les contraintes entre variables de différents modules (inter-module).

En vue d'optimiser le modèle CPPC, des variables d'évaluation doivent être ajoutées au modèle. Pour la configuration de produit, nous considèrerons uniquement le critère de prix. Nous proposons l'utilisation du terme « prix de vente » pour représenter indifféremment la notion de coût (matières premières ou composants) et la notion de prix de vente (point de vue fonctionnel/analyse de la valeur). En conséquence, des variables de prix pour chaque module (notées $sp_m$) sont ajoutées au modèle générique. Des contraintes numériques reliant ces variables aux variables de décision ou des contraintes d'agrégation entre ces variables permettent de calculer le prix total du produit. Un exemple du modèle correspondant est illustré par la figure 9.

Le modèle générique va nous servir pour définir des cas de test. Or une génération aléatoire de contraintes entre les variables n'aurait pas de sens réel ou ne correspondant à aucun cas industriel réel. Une analyse de cas typiques de configurations industrielles montre des motifs récurrents associant un petit nombre de variables de décision fortement connectées, une variable d'évaluation et des contraintes de configuration et d'évaluation appelés PCEP (Product Configuration/Evaluation Pattern). Quatre types de PCEP ont été définis et illustrés sur la figure 10. Ils correspondent à différents points de vue (physique et/ou fonctionnel) et différentes associations entre variables de décision (configuration et/ou évaluation) possibles. Le modèle générique de produit est alors défini comme un ensemble de modules composés de différents PCEP tel qu'illustré sur la figure 11.

Toujours pour éviter une génération aléatoire qui n'aurait pas de sens réel, quatre motifs de contraintes de configuration entre les valeurs de variables de décisions notés Tcp (Type of configuration pattern) ont été définis et sont illustrés sur la figure 12. Ces motifs décrivent des situations standards de relation de configuration entre des variables de décision. Ils permettent de générer les combinaisons possibles entre les valeurs des variables impliquées. Chaque motif peut être caractérisé par une densité de contrainte. La densité de contrainte est définie comme le nombre de combinaisons (tuples) interdites par rapport au nombre de combinaisons possible sans prendre en compte les contraintes. Ainsi une forte densité de contraintes suppose un faible nombre de combinaisons autorisées : plus forte est la densité de contraintes, plus le problème est contraint.

Dans le même état d'esprit, trois types de motif d'évaluation ont été définis et illustrés sur la figure 13. Un motif d'évaluation permet d'associer un prix (ou un autre critère) à une valeur pour la variable d'évaluation correspondante.

Enfin pour éviter un placement aléatoire des contraintes entre les variables de décision, entre les PCEPs et entre les modules, trois types d'architecture sont envisagés dans ce travail :

- Architecture de type modulaire (figure 15) : un grand nombre de contraintes de configuration à l'intérieur des modules avec une forte densité de contraintes et un faible nombre de contraintes entre les modules avec une faible densité de contraintes.
- Architecture de type intégrée (fgure 16) : des contraintes réparties de façon homogène à l'intérieur et entre les modules avec une densité homogène.
- Architecture de type plateforme (figure 18) : même répartition des contraintes et densités que l'architecture modulaire mais avec un module central connecté aux autres modules et pas d'autres contraintes entre les autres modules.

## 3.2 Modèle générique de configuration de Processus

Dans une vue de haut niveau, le processus d'obtention d'un produit peut être considéré comme un ensemble d'activités ou opérations configurables reliées par des relations de précédence. A ce niveau d'abstraction, le nombre d'opérations et leur enchaînement sont considérés comme statiques (pas de choix entre différentes opérations ou d'activation d'opérations selon le produit associé).

Il s'agit ici de configurer le choix et le dimensionnement des ressources associées aux opérations principales du processus. A chaque opération est associé un ensemble de ressources quantifiables permettant de réaliser la charge de travail associée à l'opération. De même que pour les familles de composants côté Produit, les ressources capables de réaliser une opération sont regroupées en familles de ressources ($for_m$) et associées à une quantité ($qtr_m$). Ces variables constituent les variables de décision du modèle générique de Processus. Ces variables sont reliées par des contraintes de configuration illustrant soit le dimensionnement (association ressource/quantité) soit les compatibilités entre ressources.

Afin d'évaluer les combinaisons ressources/quantités possibles, des variables de prix ($sp_m$) et de durée ($dur_m$) sont associées à chaque opération. L'évaluation nécessite également l'ajout d'une variable intermédiaire pour représenter la charge de travail associée à l'opération ($wl_m$). Des contraintes d'évaluation permettent alors de déterminer i) la durée d'une opération selon la charge de travail, la quantité et la ressource sélectionnée et ii) le prix de l'opération selon la ressource et la charge de travail associée. Côté coût, des contraintes numériques permettent d'agréger les coûts de chaque opération pour obtenir le coût total du processus. Enfin, ce modèle générique est complété par l'ajout de variables et de contraintes temporelles : date de début et de fin de chaque opération, les contraintes numériques associant ces dates à la durée de chaque opération et les contraintes numériques de précédence représentant le séquencement des opérations du processus. Une illustration du modèle générique de processus est donnée par la figure 21.

### 3.3 Couplage des modèles génériques Produit et Processus

Le couplage des deux modèles génériques précédents correspond à un ensemble de contraintes de configuration limitant les combinaisons possibles entre les variables de décision des deux modèles. Trois types de couplage sont envisagés. Un choix sur le produit peut i) restreindre le choix de ressources pour une opération (par exemple, un produit nécessité une compétence particulière) ; ii) nécessiter une quantité particulière (par exemple, il faut au moins deux opérateurs) ; ou iii) être associé à une charge de travail particulière. Les contraintes de couplage ne portent jamais sur les variables d'évaluation du modèle (prix et durées).

### 3.4 Caractéristiques principales du modèle générique Produit et Processus

Le modèle générique Produit/Processus complet ainsi défini permet de représenter une grande variété de cas industriels. Il fournit un cadre permettant une évaluation réaliste des algorithmes d'optimisation pour ce type de problème en évitant une génération aléatoire des variables et des contraintes de configuration et d'évaluation.

Les caractéristiques clés de ce modèle sont : i) sa taille en termes de nombres de modules, d'opérations, de variables et la taille de leurs domaines ; ii) son architecture produit (plateforme, modulaire ou intégrée) et iii) la densité et la nature des contraintes (pattern de configuration).

## 4. Benchmark et évaluation de l’approche existante

### 4.1 Benchmark de huit cas de test

Le modèle générique précédemment proposé permet de générer des instances de problème O-CPPC. Afin de tester les caractéristiques clés du problème (architecture du produit, densité de contrainte et taille du problème), nous avons généré huit cas de test présentés en annexe. La procédure pour spécifier les cas de test est illustrée par la figure 24 et un générateur de cas de test a été mis au point. Un cas de référence a été défini avec une architecture de type plateforme, une taille de 30 variables de décision et une densité de contrainte médiane (50% de tuples interdits par contraintes). Puis nous avons fait varier soit la taille (15, 60 et 100 variables), soit le type d’architecture, soit le niveau de densité de contrainte (niveau bas impliquant 20% de tuples interdits et niveau haut impliquant 80% de tuples interdits) pour obtenir un plan d’expérience avec huit cas de test résumés dans le tableau 3. Les cas de test ont été définis de façon à évaluer séparément les trois caractéristiques clés du problème (taille, architecture du produit et niveau de contraintes).

### 4.2 Evaluation de l’algorithme CFB-EA

L’algorithme CFB-EA a été utilisé pour chacun des cas de test. Ses performances sont analysées en termes de qualité de solutions fournies et de temps d’exécution. La métrique de l’hypervolume (HV), illustrée sur la figure 26, est utilisée pour évaluer à la fois la performance et la diversité des solutions trouvées. Un plan d’expérience restreint a été mis en œuvre sur le cas de référence pour sélectionner les valeurs des paramètres évolutionnaires (taille de population, de l’archive et probabilités de croisement et de mutation) qui sont ensuite utilisées pour les autres cas de test. Le temps d’exécution de l’algorithme a été défini pour chaque cas de test de façon à laisser à l’algorithme le temps de converger. Comme il s’agit d’un algorithme pseudo-aléatoire, chaque cas est testé 5 fois. Les résultats proposés sont une moyenne de ses 5 exécutions. Afin d’étudier l’évolution de la performance au cours du temps, les temps de calcul moyens nécessaires pour atteindre la performance final (notée $HV_{final}$), 99% et 99.9% de cette performance sont comparés.

La performance sur le cas de test référence est présentée par la figure 27 et le tableau associé. Elle montre que l’algorithme est efficace pour obtenir très rapidement une bonne qualité de solution (environ 10% du temps pour obtenir la performance finale) mais demande un temps conséquent pour affiner ce résultat.

La comparaison des cas de test avec des tailles différentes montre que le temps de calcul nécessaire est fortement lié à la taille du problème. Néanmoins, il reste dans des limites acceptables par rapport à la démarche d'aide à la décision envisagée (1/2 journée de calcul pour le cas à 100 variables de décision).

La comparaison des différents niveaux de densité de contraintes montre également un impact élevé sur le temps de calcul. Plus le modèle est contraint, moins il a de solutions faisables et moins le temps de calcul est élevé. Le cas le plus contraint (80% de tuples interdits) demande 10 fois moins de temps (247 sec.) pour atteindre sa performance finale que le cas le moins contraint (20% de tuples, 2610 sec.).

L'analyse des différents types d'architecture de produit montre un avantage pour l'architecture de type plateforme puis de type modulaire. L'architecture de type intégrée semble la plus difficile à optimiser. Il faut bien rappeler que les différentes architectures présentent globalement le même niveau de contraintes (même nombre de contraintes et même ratio de contraintes avec des densités forte/moyenne). La différence entre les cas réside dans la répartition des contraintes dans le modèle ou la répartition des contraintes à forte ou moyenne densité entre les modules du produit. Généralement, il semble que plus les contraintes sont réparties dans le modèle, plus l'optimisation est difficile. Dans tous les cas, l'algorithme montre une bonne capacité à trouver rapidement de bonnes solutions, mais nécessite un temps de calcul plus long pour trouver les valeurs finales.

## 5. Amélioration de l'approche existante et expérimentations

### 5.1 Amélioration proposée, CFB-EA+

L'idée à la base de ces travaux est d'utiliser la capacité de la méthode CFB-EA à fournir rapidement un front de Pareto approximatif. Ce front de Pareto peut être montré à l'utilisateur qui, une fois informé, peut affiner sa demande en formulant une préférence sur une zone d'intérêt dans l'espace de recherche (contraintes de prix et/ou de délai maximum). Une second phase d'optimisation est alors enclenchée mais en restreignant la recherche à la zone délimitée par l'utilisateur. Tout en conservant son caractère d'aide à la décision multi-objectif *a posteriori* (l'utilisateur ne donne pas de préférence entre les objectifs avant l'optimisation), cette méthode appelée CFB-EA+ permet d'accélérer le recherche de solutions faisables satisfaisant les besoins et préférences de l'utilisateur.

La restriction de la recherche à la zone choisie par l'utilisateur est réalisée en ajoutant de nouvelles contraintes sur les variables objectifs. Ainsi les solutions sont maintenues dans la zone délimitée par l'utilisateur. Cette démarche, illustrée sur la figure 31 et dont l'algorithme est donnée en figure 32, comporte deux éléments à étudier : la durée de la première phase de recherche globale (temps noté ST pour Switching Time) et les caractéristiques de la zone délimitée par les préférences de l'utilisateur. Concernant la durée de la première phase (ST), si elle est trop importante, cela limite le gain de temps de calcul total espéré ; si elle est trop courte, le front de Parteo proposé à l'utilisateur ne lui permet pas de choisir correctement une zone de préférence. Concernant la zone choisie par l'utilisateur, si elle trop petite, l'algorithme risque d'être piégé dans un optimum local (faible population initiale) ; si elle est trop large, le gain de temps de calcul risque d'être amoindris.

### 5.2 Expérimentations et recommandations

Le cas de test référence a été utilisé pour comparer les deux méthodes. Trois zones différentes (notées RA1, RA2 et RA3) ont été testées ainsi que trois durées de la première phase (respectivement le temps pour atteindre 70, 80 et 90% de la performance finale par CFB-EA).

La méthode proposée permet une réduction significative du temps de calcul de près de 25% en moyenne. L'analyse des résultats montre que le critère essentiel pour le succès de la seconde phase est la diversité des individus dans la zone choisie à l'issue de la première phase.

En conséquence, une nouvelle version de CFB-EA+ est proposée. Cette amélioration dont l'algorithme est présenté par la figure 40, ajoute comme paramètre un nombre minimal de solutions dans la zone sélectionnée par l'utilisateur. Une nouvelle série de test a été réalisée avec cette nouvelle version sur le cas de test à 60 variables. Les gains de temps de calcul obtenus sont importants avec en moyenne 54% de réduction.

## 6. Perspectives

Ces travaux proposent un modèle générique pour le problème de configuration et d'optimisation conjointe Produit/Processus, capable de représenter une grande diversité de problèmes industriels. Différentes architectures et de nombreux patterns tant sur la répartition et les liens entre variables que sur la nature des contraintes de configuration ou d'évaluation sont proposés. D'autres architectures, critères ou patterns pourraient être ajoutés. Le modèle générique proposé permet de générer des cas de tests représentatifs et éviter ainsi une génération aléatoire aberrante.

Un benchmark utilisant le modèle générique a été mis au point et a permis d'évaluer l'approche CFB-EA par rapport à trois caractéristiques clés du problème : la taille, l'architecture produit et le niveau de contraintes. Un plan d'expérience plus complet pourrait être mené pour évaluer l'impact des autres caractéristiques du modèle ou leur interaction avec les paramètres de CFB-EA.

Une nouvelle approche d'optimisation en deux étapes a été proposée et évaluée. Elle montre des gains de temps de calcul important grâce à une interaction avec l'utilisateur. Des recommandations ont été proposées pour cadrer cette interaction. Une comparaison avec d'autres méthodes utilisables pour la seconde phase telles que les SLS (stochastic local search) ou une approche exacte pourrait être envisagée.

# Introduction

In the manufacturing industry, the concept of mass customization has established itself as an indispensable lever for gaining market share. The aim is to offer a high level of diversity of product or service to the customer while maintaining a good level of productivity (Pine , 1993). In order to develop and implement mass customization, many companies use configuration software (Felfernig, et al., 2014). Configuration software enables companies to propose to their customers customized products from a huge set of variants and options of products (Wang, et al., 2015). If all configuration software can assist the supplier during the configuration of the product, only some of them do the same for the associated production and/or delivery process. Handling the two aspects (product and process) concurrently allows to avoid inconsistencies (product delicate or impossible to produce or to deliver) and give an extended view of the company response to the customer needs. This problem, considering the two aspects, is called Concurrent Product and Process Configuration (CPPC). Processing and optimizing CPPC problems are a major issue for companies producing technical products or systems either in business to business (B2B) or business to customer (B2C) situations.

A configuration software is a kind of knowledge-based system that assists the user in the configuration task. The knowledge is in fact a representation or a model of all possible product and process configuration possibilities that we call the CPPC model or the generic model. The configuration software is used to confront customer's requirements and supplier needs with the CPPC model. Configuring a product or a process corresponds to the selection of options or alternatives for the product (architecture, components or functionalities selection) and for the process (mainly resources selection and dimensioning). It thus corresponds to a decision-making problem. According to many works on the subject (Sabin & Weigel, 1998) (Zhang, 2014) (Felfernig, et al., 2014), the configuration problem can be considered as a Constraint Satisfaction Problem (CSP). In such a model, decision variables represent possible configuration choices on product and/or process. These decision variables are linked by constraints that model compatibilities and relations between decision variables. This modeling paradigm allows to use specific methods and tools to support the decision-making process.

In most configuration problems, the configured product or process must satisfy a certain number of criteria which rely on the customer's requirements (as for example: price, delivery time, performance) and/or on the objectives of the company (as for example: carbon footprint, quality, production cost). It results in a multi-criteria optimization problem named O-CPPC.

This problem is particularly difficult because of the many variants of the product and process, as well as their relationships and their impact on various objectives.

Despite the industrial interest for this type of problem, we identified a lack of benchmark (set of testing instances) to evaluate and compare optimization methods. A benchmark avoids the case-dependency of optimization scientific results. Consequently, in this thesis: (i) a generic model of CPPC is proposed in order to generate a benchmark, (ii) this model is then used to evaluate an optimization algorithm previously published by our laboratory (Pitiot, et al., 2013) and (iii) an original improvement of the previous optimization algorithm, that reduces computation time, is proposed and evaluated.

# 1. Introduction: domains, problems and tools

This PhD therefore focuses on modeling and optimizing the O-CPPC problem as well as evaluating optimization approaches. In the following sub-sections, the domains of the problem that originate the development of the present research are described. Then we introduce the frameworks and tools that we will use to model and optimize the problem. Afterward we propose the three research questions that we consider and finally we present the organization of the document with the different stages that comprise the following thesis.

## 1.1 Domains of the problem

We describe below the four domains covered by the addressed problem: Product Configuration, Process Configuration, Concurrent Product and Process Configuration (CPPC) and Optimization of the Concurrent Product and Process Configuration (O-CPPC) and we conclude with our research interest.

### 1.1.1 Product Configuration

In this section we consider the product and define relevant configuration task and problem and discuss product configuration applications and software.

#### 1.1.1.1 Product configuration Task Definition

Formally speaking, (Sabin & Weigel, 1998) define the configuration task as "a special case of design activity where the artifact being configured is assembled from instances of a fixed set of well-defined component types" which can be composed conforming a set of constraints. As mentioned by (Dhungana, et al., 2017), product configuration is a well-established methodology for generating and building individualized products. With close ideas, other authors like (Mittal & Frayman, 1989), (Soininen, et al., 1998), (Aldanondo, et al., 2008) or (Hofstedt & Schneeweiss, 2013) have defined configuration as the task of deriving the definition of a specific or customized product (through a set of properties, subassemblies or bill of materials,…) from a generic model, while taking into account specific customer requirements.

#### 1.1.1.2 Product configuration Problem Definition

We can say that the core problem of product configuration is to select and arrange combinations of parts or components that satisfy given specifications (Sabin & Weigel, 1998).

The first to propose a formal definition of the configuration problem were (Mittal & Frayman, 1989). This definition states:

- "*Given: (A) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected to that port, and other structural constraints (B) some description of the desired configuration; and (C) possibly some criteria for making optimal selections.*
- *Build: One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connections between the components in the set, or, detect inconsistencies in the requirements."*

Most of authors have been more or less adapting this definition. (Aldanondo, et al., 2008) have proposed to add to this rather physical definition (component based) a descriptive or functional view. (Felfernig, et al., 2014) mentioned that the integration of the configuration process sharing of existing components and assemblies within product family architectures, involve a number of different stakeholders and experts from various company sectors. Therefore, the configuration problem has clearly a multidisciplinary context because it involves the knowledge of many functions of a company (marketing, sale, design, production…) each contributing from its space to the subject of interest.

#### 1.1.1.3 Applications

As mentioned by (Felfernig, et al., 2014) configuration task is one of the applications of the Artificial Intelligence with more successful achievements in companies of very different sectors. For example product configuration field could be illustrated with various industrial cases: automotive (Amilhastre, et al., 2002), (Kaiser, et al., 2003), (Sinz, et al., 2003); power supply (Jensen & Lars, 2005); aircraft (Kopisch & Gunter, 1992), (Zhang , et al., 2013) or train design (Han & Lee, 2011). A database of industrial cases and related research issues was started by the University of Copenhagen (Subbarayan, 2006) but it is not any more maintained.

#### 1.1.1.4 Software

A product configurator is a tool that supports the user during the configuration process (Schierholt, 2001). They are widely used to obtain product specifications in assemble to order situations. As mentioned by (Wang & Tseng, 2012) their objective is to reduce the confusion among customer decisions inside the huge number of choices in order to generate higher level of satisfaction. The task of a conventional product configurator is to "guide a consumer through the derivation of a concrete product from the product family representation so that all requirements are fulfilled" (Dhungana, et al., 2017). (Sabin & Weigel, 1998) explained that since Digital Equipment Corporation used the R1/XCON system in 1982 to configure computer systems, a wealth of configuration expert systems has been built for configuring computers, communication networks, cars, trucks, operating systems, buildings, circuit boards, keyboards, printing presses and so forth. Product configuration software is either stand-alone software (as Tacton[1], Configit[2], Pros[3] for example) or module of ERP software (as SAP[4], Oracle[5] for example).

#### 1.1.1.5 Synthesis

In the proposed research we are going to focus on configurable technical products and will not consider service or software configuration. We consider that a configurable product corresponds to a product family with all possible options and variants (Campagna & Formisano, 2013). We will consider that a configurable product can be represented formally with a generic model relying on constraint-based approaches.

### 1.1.2 Process Configuration

In this section we consider the process and define relevant configuration task and problem and discuss process configuration applications and software.

#### 1.1.2.1 Process configuration Task Definition

Much less authors publish on process configuration; many more speak of process planning. Process planning as explained by (Schierholt, 2001) is "*the task of finding relevant processes for manufacturing a product, sequencing these processes and defining the complete set of parameters for each process*". So the process planning is therefore the activity of precisely specifying how to manufacture a product. Consequently (Schierholt, 2001)concludes that it was logical to extend the principles and key ideas of product configuration to the problem of process planning and was the first to speak of process configuration.

[1] https://www.tacton.com/
[2] https://configit.com/9
[3] https://pros.com/
[4] https://www.sap.com/products/cpq.html
[5] https://www.oracle.com/applications/ebusiness/products/configurator/

Other authors like (Bartak , et al., 2010) or (Zhang , et al., 2013) follow previous ideas and have shown that the same kind of reasoning for the product configuration can be considered for the process configuration. They therefore consider that deriving a specific production plan (operations, resources to be used, others...) from some kind of generic process plan while respecting product characteristics and customer requirements, can define the process configuration. Similarly, (Tiihonen , et al., 2014) proposed a formal definition of process configuration as the task of "*transforming a given process model into one that is specifically adapted to a given set of requirements*".

#### 1.1.2.2 Process configuration Problem Definition

As the domain of process configuration has been much less studied, as far as we know there is not any formal definition of the process configuration problem well accepted. Most authors as (Schierholt, 2001) or (Gottschalk & La Rosa, 2010) explain that the idea of product configuration can be extended towards process configuration. However, inspired by (Mittal & Frayman, 1989), (Aldanondo & Vareilles, 2008) proposed the following definition for the process configuration problem:

- *"Given:*

  *(i) a generic model of a configurable routing able to represent a family of production processes with all possible variants and options, that gathers:(1) a set of operations, (2) a set of resources with a required quantity, (3) a set of various constraints that restricts possible combinations of operations, resources and required quantities,*

  *(ii) a set of inputs, where an input corresponds with a selection of an operation, a resource or a quantity value,*
- *Routing configuring can be defined as 'finding at least one set of operations with relevant sets of pairs (resource, quantity) that satisfies all the constraints and the inputs*."

The problem of numerous stakeholders of product configuration is much less present in the process configuration because it interests mainly functions relevant to production and delivery, in other words the customer is buying a product not a process.

#### 1.1.2.3 Applications

Very few "pure" process configuration applications relevant to physical technical product cases can be found in the literature. We mean by "pure", without a strong link with product configuration application. A process configuration system including an interactive and automatic process configuration component has been developed and deployed in a Swiss metal working company (Schierholt, 2001). (Gottschalk & La Rosa, 2010) present a process configuration case dealing with a film industry. If we consider process configuration for services and software, more cases can be found. As mentioned by (Felfernig, et al., 2014) we can find success applications of process configuration in service areas like telecommunication and financial services. (Mayer, et al., 2011) described a large software and hardware development process and (Gottschalk, et al., 2009) applied a process configuration in Dutch municipalities.

#### 1.1.2.4 Software

As far as we know, there is no process configuration software that exists as a standalone software. At the opposite, most of product configuration software editors propose modules that allow configuring the process associated with the configured product.

#### 1.1.2.5 Synthesis

In the proposed research we are going to focus on discrete configurable processes and will not consider continuous processes (as chemical or energy production process). We consider that a configurable process corresponds to a discrete production process with all possible options and variants. We will consider that a configurable process can be represented formally with a generic process model relying on constraint-based approaches. Key difference with product, most of the times the customer doesn't care about the process configuration which is most of the times considered as a "pure" supplier problem. The point that can interest the customer is the delivery date but not the way the supplier reaches it.

### 1.1.3 Concurrent Product and Process Configuration (CPPC)

Here too, CPPC task and problem definitions are proposed. Then applications and software are discussed.

#### 1.1.3.1 CPPC Task Definition

Most of the time, a product is first configured and then once the product specificities are decided, process configuration is launched, in other words: begin with "the what" then finish with "the how". The key interest of breaking this sequence is to allow taking into account specific process requirements before product requirements (Zhang , et al., 2013) or (Baxter, 2007). For example, in some situations the use of a specific resource may forbid a specific product component or a strong due date expectation may oblige to use a local expensive resource. These resource selections can have strong consequences on the product configuration.

We can consequently define the Concurrent Product and Process Configuration as the task of configuring a product and its related process without strong precedence constraint.

### 1.1.3.2 CPPC Problem Definition

Most of the academic works deal with these two problems in an independent way, either product configuration or process configuration. Some studies have showed the interest of the union of these two configuration problems (Baxter, 2007), (Aldanondo, et al., 2010), (Hong, et al., 2010), (Li, et al., 2006) and (Huang & Gu, 2006). (Dhungana, et al., 2017) propose that the core problem to solve is "*to integrate configuration process in order to simplify the value chain from product configuration to manufacturing of the individualized product*". The CPPC problem can therefore be defined as the strict union of the product and process configuration problems with the addition of coupling interrelations of constraints. Coupling interrelations proposed in (Aldanondo, et al., 2010), (Pitiot, et al., 2013) formally describe the interdependences that exist between both problems. The CPPC problem of course inherits of all characteristics of product and process configuration problems.

The resulting generic model architecture is shown in Figure 1, where: (i) X are configuration variables (ii) dotted lines are interrelations or constraints. The model gathers two sets of variables (black rounded boxes in Figure 1), one associated with the product generic model and the other with the process generic model. Constraints or interrelations are specific either to product or process (lower part of Figure 1) or between both of them (previous coupling interrelations in the upper part of Figure 1).

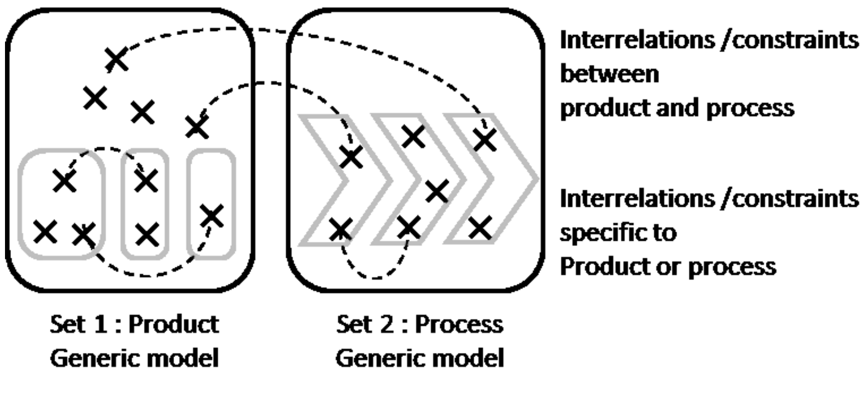

Interrelations /constraints between product and process
Interrelations /constraints specific to Product or process
Set 1 : Product Generic model
Set 2 : Process Generic model

*Figure 1- Generic model architecture of the CPPC problem*

### 1.1.3.3 Applications

(Dhungana, et al., 2017) proposed novel concepts and algorithms for a holistic approach in order to integrate product and production configuration. For their pilot study they used a power controller module as a configurable product and its production requirements. (Zhang , et al.,

2013) show an example of CPPC problem dealing with small private aircraft. (Aldanondo, et al., 2010) illustrate this problem with an example dealing with crane. Other product examples mentioned in their model are the laptops, bicycles and power supplies.

#### 1.1.3.4 Software

A decision aiding tool has to assist stakeholder to make the best decisions (product configuration and process planning choices) according to multiple objectives. CPPC problems take place in the first steps of the study of product and associated process. (Campagna & Formisano, 2013) presented a framework called ProdDoc, who combined Product and Process Configuration. As mentioned by (Dhungana, et al., 2017), ProdDoc is a step towards considering manufacturing during product configuration, because it provides constraint based languages for both product modeling and specification of process steps for production. At the present time we cannot strongly comment about the ability of commercial configuration software to handle simultaneously product and process configuration.

#### 1.1.3.5 Synthesis

Being consistent with the previous sections, we are going to concentrate on problems assembling concurrently product and process configuration. We will restrict our investigations on: (i) technical products and systems, so we don’t deal with services and software, (ii) discrete manufacturing process, so we don’t consider chemical or continuous food industry processes. We will assume that the resulting product-process generic model of the configuration problem can be considered as a constraint satisfaction problem.

### 1.1.4 Optimization of Concurrent Product and Process Configuration (O-CPPC)

In this last sub section, as the scope is rather narrow, we just propose definition elements for the CPPC optimization task and problem.

#### 1.1.4.1 CPPC Optimization task definition

The configuration task can be achieved either autonomously or interactively. By autonomous configuration, we mean that the customer provides all the requirements in a single shot and then ask for a solution. By interactive configuration, we mean that after inputting each “elementary requirement” provided by the customer, the consequences on other configuration variables are computed and shown to the user. By “elementary requirement”, we mean a customer domain restriction of a single product or process configuration variable (for example: required power belongs to [6, 8] or final assembly operation resource should be “Asia-line”).

As the goal of companies using configuration techniques is to propose a wide range of possibilities, the quantity of configuration variables that should be valuated can be rather large and the relevant interactive configuration process rather long.

In order to avoid asking the customer to input elementary requirements on all variables, it is necessary to be able to switch the configuration process at any time from the previous interactive mode towards the autonomous mode. This means that once all elementary requirements proposed by the customer have been sequentially processed, remaining undecided choices are processed thanks to some autonomous computations. This autonomous computation can be achieved either with default values or by using multi-criteria optimization (cost, due-date, performance, carbon footprint, etc.).

This thesis is dedicated to the second solution and for the sake of clarity we will assume only two conflicting criteria: cost and cycle time. Furthermore, instead of providing a single solution, we propose a Pareto front so that the customer can choose, according to preference criteria, a rather low delivery time (at a higher cost) or a rather low-cost solution (with a longer delivery time). Therefore, the process of interactive configuration and optimization of the CPPC behaves as a two-step process, which combines interactive Concurrent Product and Process Configuration in step 1 and autonomous Concurrent Product and Process Optimization in step 2, as shown in Figure 2.

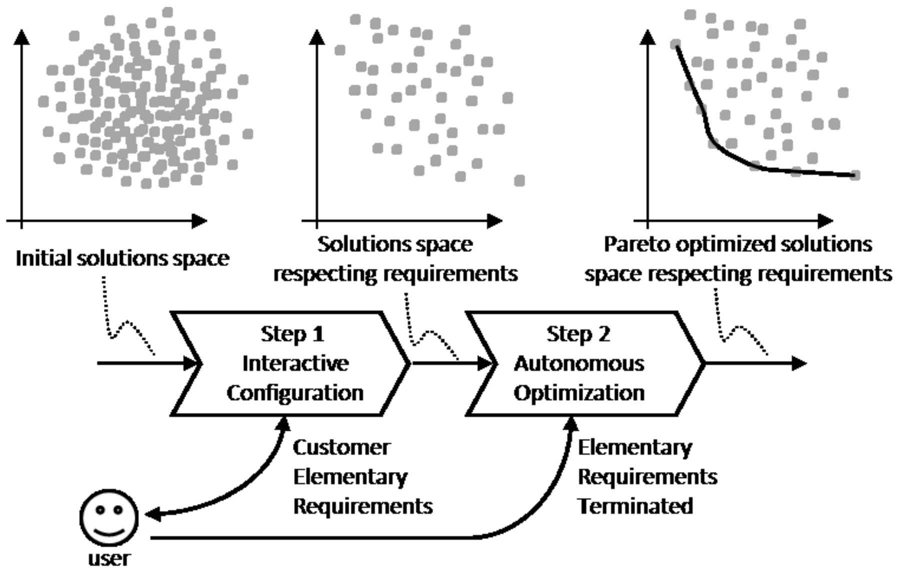

Initial solutions space
Solutions space respecting requirements
Pareto optimized solutions space respecting requirements
Step 1 Interactive Configuration
Step 2 Autonomous Optimization
Customer Elementary Requirements
Elementary Requirements Terminated
user

*Figure 2- CCPC configuration and optimization process*

1.1.4.2 CPPC Optimization Problem Definition

Most of the works that have been dealing with configuration optimization present this problem as a constrained optimization problem (Li, et al., 2006), (Wei, et al., 2014), (Du, et al., 2014). Our CCPC optimization problem inherits of this characteristic. The only difference with the CPPC problem lies in the existence of optimization criteria. A criterion can depend of configuration variables of product, process or both. A technical performance criterion may depend of product variables, a cycle time or delivery date criteria may depend of process, while a cost criterion may depend of both. For a two criteria problem (cycle time and cost), the resulting generic model architecture is shown in Figure 3 where criteria dependence has been added to the CPPC problem generic model of figure 2.

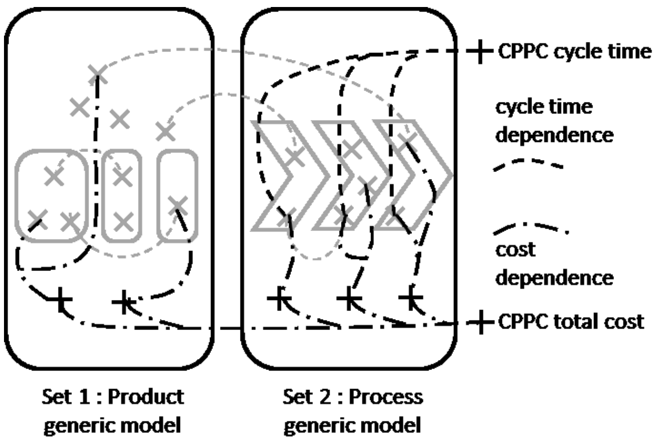


*Figure 3- Generic model architecture of the CPPC optimization problem*

1.1.4.3 Synthesis and application issues

According to previous sections, we will consider in the present research the problem of Optimization of the Concurrent Product and Process Configuration. We assume or consider (i) only technical product and discrete production process, (ii) only situation where optimization follows interactive configuration, (iii) only two optimization criteria cost and cycle time.

Given the problem under study and our concern with optimization, it is important to mention practical industrial or application issues. These issues may be very different, since configuration situations may range: (i) from cases costing around a thousand euros up to situations involving millions of euros, (ii) from business relations that are either business to customer (B2C) or to business (B2B).

In B2C, configuration techniques can be used to configure: personal computers costing 1-4 k€, kitchens ranging from 10-20 k€, cars from 20-80 k€, or sailing boats from 200-800 k€. For these B2C situations, it is usually clear that if the customer can wait a day for some optimization to a boat, he will hope to get results in just a few hours for a car and in less than an hour for a kitchen or a computer. As this kind of selling is frequently achieved face to face, with customers that are not fully driven by rational concerns, solutions optimization with many “what if” issues requires computation times that should be as low as possible in order to meet any kind of customer request almost on demand.

The same kind of conclusion can be drawn for B2B situations, which may involve, for example, machine-tools with a value of 50-200 k€, cranes costing 200-800 k€, private planes ranging from 2-10 M€ or a plant facility of 5-20 M€. For these B2B situations, given the high cost and the fact that customer demand is rather more rational, it is not a problem to wait one day for optimization results. But the concern here is more on the optimization side, which can require effective optimization, meaning for example that if a 0.1 % energy efficiency bonus can be achieved, the optimization process should find it.

Consequently; these two needs drive expectations of low computation times and solution optimality which are the core of this research.

## 1.2 Overview of frameworks and tools

We just introduce in this section the two used frameworks: Constraint Satisfaction Problem (CSP) and Evolutionary Algorithms, in order to be able to propose research questions and a manuscript organization.

### 1.2.1 CPPC as a Constraint Satisfaction Problem (CSP)

Various modeling and processing approaches can be used to deal with the configuration problem. The 6th chapter of the configuration book (Felfernig, et al., 2014) identify and compare various approaches that can handle the configuration problem: rule-based systems, constraint satisfaction problem (CSP), Sat Solving, feature models, Unified modeling language configuration models, Description Logics, Ontology, Answer Set Programming and "hybrid" methods that assemble two of these methods.

Historically, first configuration software or systems were rule based (McDermott, 1982). As the maintenance of such system becomes very quickly intractable (Soloway , et al., 1987), most of researchers have been working on approaches that differentiate the product knowledge domain form the problem solving knowledge. In this idea CSP techniques strongly supported by (Freuder, 1997) state: "*Constraint technologies are one of the closest approaches computer science has yet made to the Holy Grail of programming: a user states the problem, the computer solves it.*" The clear distinction of modeling and solving offered by CSP approaches and the importance of constraints or interrelations between configuration variables are some of the key reasons why CSP approaches are strongly used by configuration authors. A drawback of CSP is their poor ability to deal with problem structuring aspects which is important for both product configuration (a product is a set of sub-assemblies; a sub-assembly is a set of components…) and process configuration (a process is a set of operations; an operation is a set of tasks…). This is why other techniques, like UML, Description Logics or Ontology (dealing very well with hierarchical aspects), are also used in configuration but they are almost always associated with some rules or constraint approaches in configuration.

Furthermore, if constraint based approaches fit configuration well, they also fit very well process planning (Bartak , et al., 2010) and optimization problems (Coello, s.f.). As a consequence, we will use this constraint satisfaction problem framework (CSP) in this work. The first section of the second chapter will propose a survey of CSP and configuration.

### 1.2.2 Optimization of CPPC with Evolutionary Algorithms (EA)

With respect to optimization the CPPC problem inherits specificities from both product configuration optimization and process planning optimization.

A first specificity of this is that the solution space can be large. It is reported in (Amilhastre, et al., 2002) that a configuration solution space of more than $1.4 \times 10^{12}$ is required for a car-configuration problem. When planning is added, the combinatorial structure can become very large. (Pitiot, et al., 2014) investigated solution spaces range between $10^6$ and $10^{17}$ are investigated.

A second specificity is that the CPPC solution space may vary in terms of size but also in terms of position. We have introduced in section 1.1.4 the notion of elementary requirements that correspond to customer expectations that must be respected in opposition with undecided choices that can be set by the optimization process. Therefore, for the first point, it is important to note that the size of the problem we are seeking to optimize increases when the quantity of elementary requirements processed during interactive configuration decreases. For the second point, according to the content of these elementary requirements (either high or low product performance, for example, that essentially drives cost and cycle time), the corresponding solution space that needs to be optimized can be clearly located in a different space area. This is shown in Figure 4, where two kinds of elementary requirements drive very different solution space locations, high performances elementary requirements drive costly and long cycle time while it is the opposite for low performance product elementary requirements.

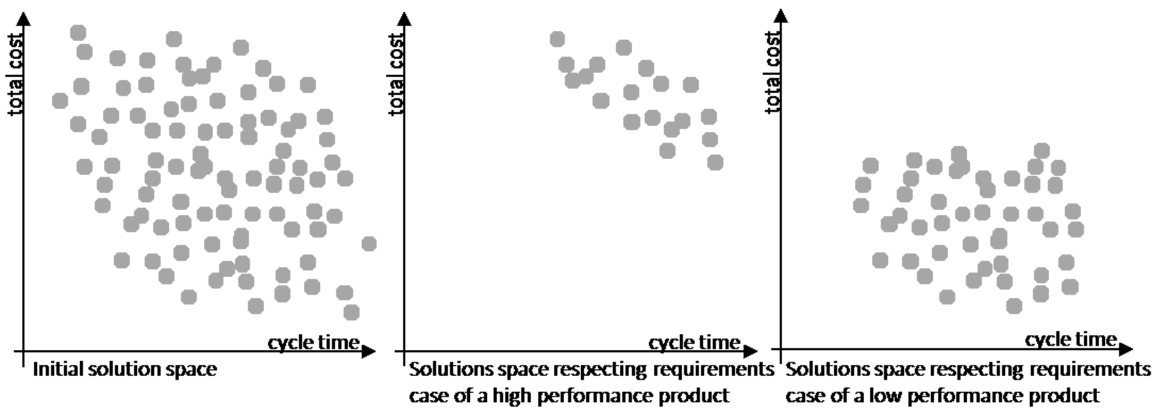


*Figure 4- CPPC optimization: different solution space locations*

A third specificity is relevant to the constraint level. By constraint level we mean the ratio between feasible and unfeasible space that can be computed as follows. We first consider the number of configuration variables (*n*) where the customer can express an elementary requirement and the average number of possible values (*p*) for each. When constraints are not considered, the solution space size equals $p^n$. When constraints are considered, the solution space is reduced with respect to a constraint level which is the ratio of the number of constrained solutions (*c*) divided by the number of unconstrained solutions (*u*). This level can be quantified for a single constraint or for a whole problem. One should be aware that a high level corresponds to a low constrained problem. Thus, constrained solution space size equals $p^n$ * c/u. As the goal of companies using configuration techniques is most of the time to propose as many solutions as possible, the CPPC problem is most of the times not over-constrained.

The last specificity deals with the number of criteria that must be taken into account. We have already mentioned, that we will consider only the two-criteria cycle time and cost that are the most frequent criteria required by customer. However, assuming that technical performance has been taken into account with elementary requirements, other criteria like product and process quality level and/or carbon footprint could be added. Consequently, for our CPPC optimization problem the number of criteria is rather low and, in any case, lower than five. As already said, we will consider only two criteria but, in some situations, we will discuss problems with more than two criteria.

Given these specificities, we have tried to find the main authors close to our CPPC problem optimization. We therefore made a query on the web of science with following title words: "product” and “configuration” and “optimization” and not “supply chain”. As a result the ten most cited papers were in citation decreasing order: (Li, et al., 2006), (Du, et al., 2014), (Zhou, et al., 2008), (Chen & Lin, 2002), (Pitiot, et al., 2014), (Xu, 2005), (Viswanathan & Allada, 2006), (Wei, et al., 2014), (Jiang, et al., 2011) and (Song & Chan, 2015). Among these ten papers, eight where running genetic or evolutionary approaches while the two remaining were based on game theory or pairwise comparisons. We can therefore conclude that these meta-heuristics fit very well the CPPC optimization specificities.

As a consequence, we will use an evolutionary approach (EA) in this work. The second section of the second chapter will detail a survey of various EA.

## 1.3 Research Questions and manuscript organization

Having described the domains and introduced the two used frameworks and tools (CSP and EA) we can know define our research questions and the resulting manuscript organization.

### 1.3.1 Goals of the works

We deal with the problems of Optimization of Concurrent Product and Process Configuration (O-CPPC). As we have presented in the previous sections, many academic works show diverse investigation cases associated with product and/or process configuration. Most of the times the configuration approach is achieved with an individual focus that doesn´t take into account that there is a close relationship between a product and their respective production process. In the scientific literature, some studies tackle the analysis of Concurrent Product and Process Configuration problems. For example: (Dhungana, et al., 2017), (Campagna & Formisano, 2013), (Baxter, 2007), (Zhang , et al., 2013), (Hong, et al., 2010), (Li, et al., 2006) and (Huang & Gu, 2006); as far as we know, no paper deals with its optimization. Considering the ten articles studying configuration optimization mentioned at the end of section 1.2.2, all of them only deal with product configuration. Furthermore, most of them are rather theoretical propositions with an evaluation running a single problem, a kind of "toy" problem in most situations mainly to explain and illustrate the concepts (computer, electric motor, plane, air compressor, gear train…). There isn't any kind of design of experiments for the proposed optimization approach. As a consequence, there is no standard to analyze and compare the optimization performance and therefore most of published results can be more or less considered as case dependent.

In a previous work (Pitiot, et al., 2013), the Concurrent Product and Process Configuration (CPPC) is considered as a constraint satisfaction problem (CSP). In this work, an optimization metaheuristic, called Constraints Filtering Based - Evolutionary Algorithm (CFB-EA), has been developed and evaluated with a specific small aircraft configuration problem. The case dependency problem has driven the authors to enlarge the previous aircraft example with a design of experiments with different problem sizes and different constraint levels (Pitiot, et al., 2014).

Following this stream of works, the goals of the PhD are: (i) to propose and discuss a kind of CPPC generic model that can reduce case dependency when evaluating optimization methods, (ii) to evaluate CFB-EA with respect to this CPPC generic model and (iii) to propose and evaluate a new version of CFB-EA that reduces computation time significantly.

### 1.3.2 Research Questions

Given previous considerations, we propose the three following research questions:

QR1: Is it possible to propose a generic model of the CPPC problem that can avoid case dependency when evaluating and comparing optimization methods?

QR2: How sensitive is CFB-EA optimization method, with respect to each key characteristic of the generic model of the CPPC problem?

QR3: Is it possible to reduce the computation times of CFB-EA and other conventional EA approaches?

### 1.3.3 Thesis organization

Our main objective is therefore to evaluate and improve combinatorial optimization techniques for the problem of Concurrent Product and Process Configuration (O-CPPC) while avoiding case dependency. The rest of the thesis is organized as follows:

Research problem: Given the CPPC problem already introduced, we refine in **chapter 2** our research problem through a survey of the Constraint Satisfaction Problem with respect to configuration problems and Evolutionary Algorithms. This survey ends with the identification of the modeling and optimization approaches and tools that will be used in chapters 3 and 4.

Generic modeling: we define in **chapter 3** a generic problem of “Concurrent Product and Process Configuration”, we identify its main key characteristics and we formalize it as a constraint satisfaction problem (CSP). This generic model is the core of a problem generator that allows generating CPPC benchmarks.

Evaluation: Given a CPPC benchmark, a design of experiment is proposed in **chapter 4** to evaluate the CFB-EA performance and to validate the CPPC generic model.

Improvement: We propose and test an original improvement of CFB-EA and discuss its generalization to other evolutionary approaches in **chapter 5**.

At the end, **chapter 6** synthetizes the contributions and discusses future works.

# 2. Modeling and optimization approaches for CPPC

In this chapter, our research problem is formalized with constraints satisfaction problem (CSP) for modeling and evolutionary algorithms (EA) for optimizing. The goal is to discuss, the kind of CSP and the kind of EA which can be used in chapters 3 and 4. Firstly, we detail the CSP framework; then we recall how CSP can be used to formalize product, process, concurrent product/process configuration (CPPC) and to optimize CPPC. Secondly, we analyze genetic approaches and more particularly evolutionary algorithm with respect to our CPPC optimization goals. Thirdly, we present the confirmation of the thesis problem and highlight the most important propositions.

## 2.1 Constraint Satisfaction Problems (CSP) for configuration

We first introduce the basics of CSP, then how CSP can be used to model: product configuration, process configuration and concurrent product/process configuration.

### 2.1.1 Foundation of Constraint Satisfaction Problems (CSP)

In this section, we present constraints satisfaction problems that allow us to formalize the general knowledge necessary for the configuration activity. In a first step, we formally define the problems of constraint satisfaction and specify the types and natures of variables and constraints that they cover. In a second step, we review the different types of constraint satisfaction problems: we classify them according to (1) the type and nature of their constraints and (2) the relevance of variables in the problem and the solution.

#### 2.1.1.1 CSP framework

Constraint programming is a programming paradigm in which constraints are used to state or define the relationships between variables, and thus restrict the solution space. Constraint Satisfaction Problem (CSP) provides a unifying framework in which it is possible to express, in a natural way, a wide variety of computational problems (Krokhin & Zivny, 2017). Constraint Satisfaction Problems (CSP) allow modeling knowledge and reasoning on it to find all consistent solutions. The standard formalism of the constraint satisfaction problem goes back to (Montanari, 1974).

- **Definition 1: Constraint Satisfaction Problem**

Formally speaking, (Tsang, 1993) defines the concept of constraint satisfaction problem as follows: "CSP is a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take."

The constraint satisfaction problem consists of a triplet (X, D, C) defined by: i) a set of variables X = {x1,…,xn}, ii) a domain for each variable D = {D1,…Dn} and iii) a set of constraints C = {C1,…,Cn} linking the variables X and representing the solution space (Felfernig, et al., 2014).

- **Definition 2: Task in a CSP**

"The task in a CSP is to assign a value vk to each variable xi such that all the constraints Cj are satisfied simultaneously" (Tsang, 1993).

- **Definition 3: Solution of a CSP**

"A solution S of a CSP (X, D, C) is a complete instantiation of the variables in X satisfying all the constraints in C" (Bartak , et al., 2010).

- **Definition 4: Consistent CSP**

"If a CSP has at least one solution, it is said that the CSP is satisfiable or consistent, otherwise we say that it is inconsistent" (Bartak , et al., 2010).

- **Definition 5: Domain of a Variable**

"The domain of a variable xi is a set of all possible values that can be assigned to the variable. If "x" is a variable, then we use "Dx" to denote the domain of it" (Tsang, 1993).

- **Definition 6: Degree of a Variable**

"The variable degree corresponds to the number of constraints in which the variable is involved" (Ghédira , 2013).

- **Definition 7: Label**

A label "is a variable-value pair that represents the assignment of the value to the variable" (Tsang, 1993). The expression <x, v> is used to represent the label of assigning the value "v" to the variable "x" (Tsang, 1993) .

- **Definition 8: Constraint**

"A constraint on a set of variables is a restriction on the combination of values that these variables can take simultaneously" (Ghédira , 2013). In other words, a constraint can be seen as a set of all the legal compound labels for the subject variables (Tsang, 1993). They can be presented in different ways like functions, inequalities, matrices, etc. (Tsang, 1993)

- **Definition 9: Arity of a Constraint**

"The arity of a constraint "C" is the number of variables involved in "C"" (Ghédira , 2013). A constraint is called unary if it relates to a single variable or binary if its arity is equal to two. More generally, a constraint is called n-ary when its arity is equal to "n" (Ghédira , 2013).

### 2.1.1.2 Variables and Constraints

The variables on which the constraints apply can be of different types: symbolic or numerical, discrete or continuous. There are two ways to map the set of variables (Vareilles, 2015). The Figure 5 presents this mapping according to the type of the elements of the domains and their cardinal.

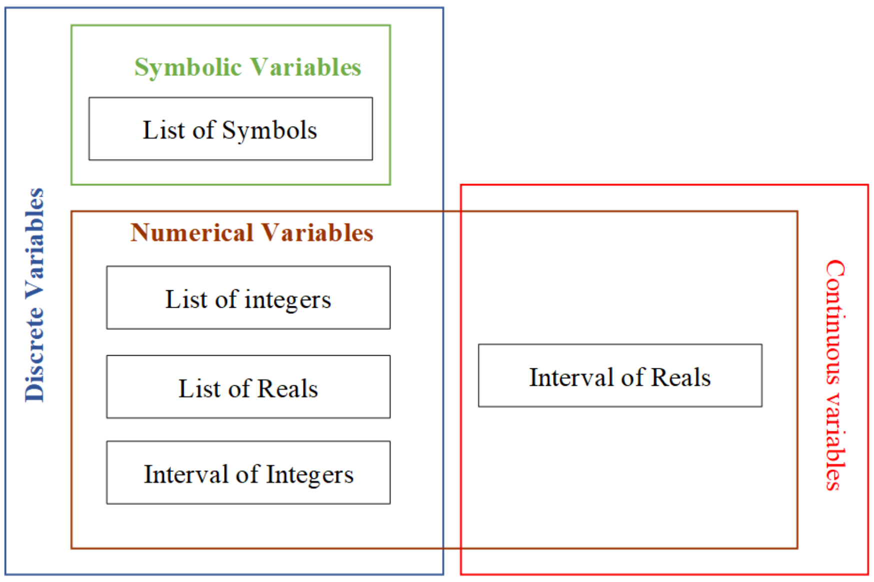


*Figure 5- Classification of variables (Vareilles, 2015)*

If we consider only the type of elements of the domains definition, we obtain two disjoint subsets: symbolic variables (in green) and numerical variables (in brown). If we consider the cardinality of the domains definition (countable or not), we obtain two other disjoint subsets: the discrete variables (in blue) and the continuous variables (in red).

On the first hand, the constraints allow limiting the space of solution by delimiting the combinations of values that the variables can take simultaneously (in these cases they are qualified as compatible). On the other hand, they allow modifying the structure of the solution space (or CSP) by adding or removing elements (variables and constraints) to the current problem to be solved (in this case they are called activation) (Mittal & Falkenhainer, 1990). The activation constraints make possible to manage the relevance of the elements (variables and constraints) of the problem by a mechanism of implicit activation or de-activation (Van Oudenhove de Saint Gery , 2006).

The diversity of knowledge to be formalized leads to the definition of different types of constraints (Yannou, 1998): a) compatibility tables representing in tabular form, the explicit list of authorized values to take into account, for instance, the allowed combinations of components, b) numerical functions representing in a mathematical form, the implicit combination of variable values to formalize, for instance, the computation of a product weight or a temporal relationship between activities or c) even pairwise functions defined by parts, allowing to take into account empirical knowledge or experimental results formalized in form of abacus (Mulyanto , 2002) (Vareilles , 2005) (Chenouard, 2007).

Figure 6 summarizes the classification of constraints, according to their nature (compatibility or activation) and their type (compatibility tables, numerical functions and charts).

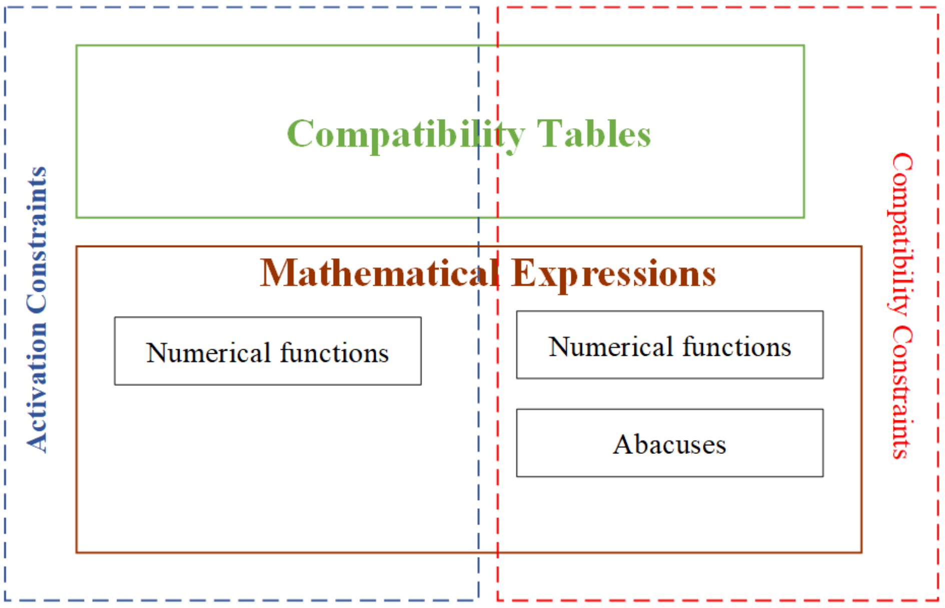


*Figure 6- Classification of constraints (Vareilles, 2015)*

### 2.1.1.3 Classification of CSP

Constraint Satisfaction Problems can be classified according to several criteria: a) the type of variables they possess (symbolic, continuous, temporal ...) (Gelle & Faltings, 2003), b) the type of the constraints (list of authorized values, mathematical function or temporal relationship) (Vareilles, 2015), and c) the nature of the established constraints (compatibility constraint and activation constraint) (Djefel , 2010) (Felfernig, et al., 2014).

(Vareilles, 2015) proposed to complete the classification of (Djefel , 2010) (Felfernig, et al., 2014) for the notion of relevance of variables in the solution. For this, the notions of CSP with static structure and with dynamic structure are stated. CSP with a static structure are CSPs where the totality of variables "X" and the totality of constraints "C" characterize the solutions. No variables or constraint can be added as the problem is solved. CSPs with a dynamic structure are CSPs where the structure can be modified by adding variables and supplementary constraint as the problem is solved.

In a similar way, we can distinguish the solutions with static structure that are described by the set of variables of "X" of the CSP. On the other hand, the solutions with a dynamic structure, contain only a subset of the variables of X, whereas these are present in the CSP, or have their structure evolve over time by adding or removing viable variables, in the same way as the CSP

Three cases are then differentiated, the fourth combination (static CSP and dynamic solution) having no meaning, as illustrated in figure 7:

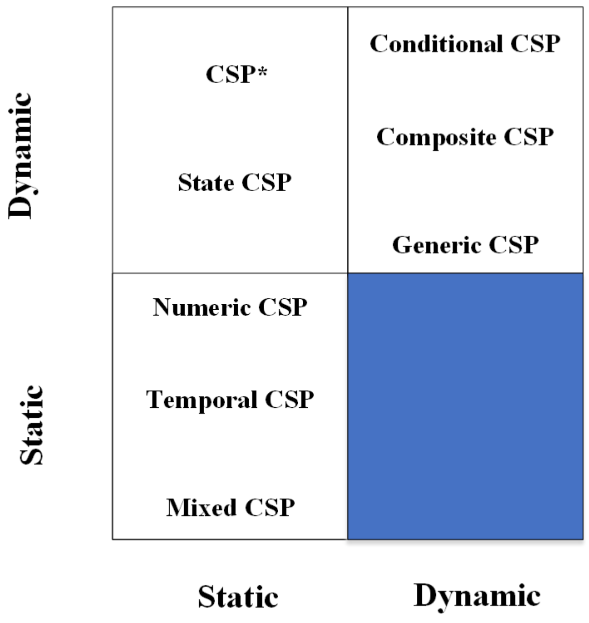


*Figure 7- Classification of CSP (Vareilles, 2015)*

Static CSP and static solution

- Discrete CSP: first type of CSP, defined by (Montanari, 1974) and characterized by discrete variables and compatibility constraints; described by lists of combinations of allowed or forbidden values (Tsang, 1993) and by discrete mathematical expressions.
- CSP continuous: extension of discrete CSPs to the continuous domain and characterized by continuous numerical variables and constraints; described in general terms as mathematical functions, and more rarely by continuous compatibility constraints.
- CSP Qualitative or quantitative temporal variables: characterized by temporal variables respectively representing time intervals (Allen, 1983); being moments or events (Dechter & Mairi, 1991) and by temporal constraints representing temporal relations between intervals or instants. (Meiri, 1996) proposed to combine these two types of temporal CSP to argue their expressiveness.
- Mixed CSPs: characterized by variables of different types and discrete, continuous, temporal or mixed compatibility constraints (Gelle & Weigel, 1995), (Vareilles , 2005).

Static CSP and dynamic solution

- CSP *: characterized by discrete variables that have, for some, the value * in their domain (optional variables) and compatibility constraints taking into account the specific value * (Amilhastre, 1999) (Macdonald & Prosser, 2002).
- State CSPs: characterized by discrete variables associated; for some, with a state Boolean variable indicating the relevance of the variable in the solution and compatibility constraints taking into account these state variables (Veron, 2001).

Dynamic CSP and dynamic solution

- Conditional CSP: characterized by discrete variables that are active or inactive in the problem, and constraints (compatibility constraints and activation constraints) that manage the relevance of variables in the problem, by explicitly allowing or prohibiting their activation according to four types of activation constraints (Mittal & Falkenhainer, 1990). Conditional CSPs have been extended to numerical variables (Gelle & Weigel, 1995), to the activation of subsets of variables (Soininen & Niemela , 1999), to the explicit activation of constraints (Vareilles , 2005) and to temporal CSPs (Tsamardinos , et al., 2003), (Vilim, et al., 2004), (Mouhoub & Sukpan, 2005).
- Composite CSP: characterized by discrete variables, some of which are meta-variables that can be substituted by an entire sub-problem (variables and constraints) and compatibility

and activation constraints (Sabin & Freuder, 1996). Composite CSPs have been introduced mainly to model the hierarchical structure of configuration problems and have been extended to temporal CSP by (Mouhoub & Sukpan, 2005).

- Generic CSP: dedicated to the configuration of products, characterized by discrete variables representing properties, ports and components, linked together by generic constraints of compatibility or activation. GCSP are by nature, hierarchical (Stumptner , et al., 1998).

### 2.1.2 Product Configuration Definition and CSP Model

In this section we present the basic definitions and a summary of the configuration techniques related to the product domain.

#### 2.1.2.1 Basic Definitions

- **Definition 10: Product**

A Product is “any good or service produced for sale, barter or internal use” (Blackstone, 2013). It has a combination of tangible and intangible attributes that an enterprise offers to a customer for purchase. A product seeks to serve a need or satisfy a want (Blackstone, 2013).

- **Definition 11: Product Architecture**

Product architecture can be defined as “the way in which the functional elements of a product are arranged into physical units and the way in which these units interact” (Jiao, et al., 2007) (Ulrich & Eppinger, 1995). For our research, we are mainly interested in integrated, modular and platform architectures.

- **Definition 12: Configurable Products**

We will focus on one type of product usually called configurable products. They have a predefined basic structure that can be customized by combining a series of available components and options (modules, parts,….) or by specifying suitable parameters (lengths, tensions,….) (Campagna & Formisano, 2013). Actually, a configurable product does not correspond to a specific physical object, but identifies a set of (physical) objects that a company can produce.

A classical definition of Configurable Product was presented by (Veron, et al., 1999) as "a set of attributes (or components) which possible values belong to a finite set, and a set of feasibility constraints over these attributes which specify their compatible combinations of values".

- **Definition 13: Product Configuration**

Product configuration is a consolidated methodology and one of the most effective technologies of mass customization strategies (Felfernig, et al., 2014) (Sabin & Weigel, 1998). It is a widely used technology for generating and building individualized products (product family design) which has been deployed by many companies for years (Wang, et al., 2015).

Generally speaking, the task of product configuration is "to search the predefined components set according to customers' requirements and the constraint relationships among the components, and to obtain the configuration results which can meet the customer's personalized requirements " (Wang, et al., 2015) (Brown, 1998).

(Aldanondo & Vareilles, 2008) summarized the following Product Configuration definition based on a compilation of common features proposed by various authors concerning product configuration (Mittal & Frayman, 1989) (Sabin & Weigel, 1998) (Soininen, et al., 1998) (Aldanondo, et al., 2003):

- Hypothesis: a product is a set of components,
- Given:

  i. a generic model of a configurable product able to represent a family of products with all possible variants and options, that gathers:

  1) a set of component groups and relevant component quantities,

  2) a set of various constraints that restricts possible combinations of components and/or component quantity values,

  ii. a set of customer requirements, where a requirement corresponds with a selection of a component or a quantity of this component,
- Product Configuration can be defined as 'finding at least one set of components that satisfies all constraints and customer requirements'.

Then the same authors proposed a lightly modification of this basic definition with the introduction of the notion of product properties, which allows to characterize the requirements and to introduce the description view as follow:

- Hypothesis: a product is a set of components,
- Given:

  i- a generic model of a configurable product able to represent a family of products with all possible variants and options, that gathers:

1) a set of component groups,

2) a set of product properties,

3) a set of various constraints that restricts possible combinations of components and/or property values

ii- a set of customer requirements, where a requirement corresponds with a selection of a component or a property value,

- Requirements and Product Configuring can be defined as ‘finding at least one set of components that satisfies all the constraints and the customer requirements'.

The component quantity introduced in the first definition by (Aldanondo & Vareilles, 2008) was considered in the last definition as a product property. The authors clarified that component quantity is mainly a physical characteristic of the product but is also necessary to consider it as a requirement (Aldanondo & Vareilles, 2008).

The configuration result is a set of components or a bill-of-materials (Aldanondo & Vareilles, 2008). In other words, we get a listing of all sub-assemblies, intermediates, parts and raw materials that go into a parent assembly showing the quantity of each required to make an assembly (Blackstone, 2013).

#### 2.1.2.2 Configuration Techniques

(Brown, 1998) summarized the next variety of techniques that can be used together to support configuring task:

Component Choice

(Brown, 1998) explained that "Component Choice" plays an important role on its usefulness. For example, the author said that a large and complex component has more requirements than others that also need to be included, so they have less flexible use. The simple and small components will probably provide more flexibility and will require more configuring; large components can be considered to be preconfigured sets of smaller components (Brown, 1998).

Experience and Knowledge

(Brown, 1998) proposed that the "Experience and Knowledge" affects the search for a configuration process. He clarifies that they can reduce the search because the knowledge allows us to build previous structured descriptions of the available components; then the experience may reduce the errors because it can allow us to build previously discovered sub-configurations or heuristic into the system (Brown, 1998).

Hierarchies

(Brown, 1998) affirms that hierarchies are abstractions that allow a descending configuration strategy in order to avoid the excessive combination generated when a lot of details at the beginning are considering. For example a "Component Hierarchy" groups specific components into types and subtypes, a "Functional Hierarchy" provides a way of storing functions organized by type and abstractness and a "Part-Subpart Hierarchy" can be used for functions, for components, or both (Brown, 1998).

Templates

(Brown, 1998) mentioned that configuration "Templates" refer to any preformed piece of configuration from past experience. For example, a template can associate functional and/or structural items and they include components and relationships between them at some level of abstraction (Brown, 1998).

Key Components

Key Components correspond to "those that are almost always required, or those on which many other choices depend, suggesting that their correct choice should take priority" (Mittal & Frayman, 1989).

Constraints

As mentioned by (Wang, et al., 2015) various valid methods for solving product configuration have been studied and constraint satisfaction problem (CSP) is one of them (Felfernig, et al., 2014). In many works on the subject (Pitiot, et al., 2014) (Sabin & Weigel, 1998) (Zhang, 2014), the Constraint Satisfaction Problem framework (CSP) has been efficiently used to model and support the product configuration activity.

Selection of components introduces new variables and new constraints (Mittal & Falkenhainer, 1990), so the constraints are introduced by decisions (Brown, 1998). The objective is to find a feasible product that satisfies not only the constraints but also the user's requirements (Veron, et al., 1999). According to this objective of choosing a feasible instance of a product among all its variations, the configuration problem can be mapped into a constraint problem. And the Constraint Satisfaction Problem (CSP) offers a suitable framework (Veron, et al., 1999).

### 2.1.3 Process configuration Definition and CSP Model

In the following section we present basic definitions and a summary of the process configuration techniques.

#### 2.1.3.1 Basic Definitions

- **Definition 14: Process**

A process is "a planned series of actions or operations (e.g., mechanical, electrical, chemical, inspection, test …) that advances a material or procedure from one stage or completion to another" (Blackstone, 2013).

Different types of processes for the products from product families are presented as follows (Wang, et al., 2015) (Schierholt, 2001):

- Standard process: the processes necessary to manufacture the product are the same, appear in the same sequence and have fixed process parameters.
- Standard process sequence with variable parameters: The processes necessary to manufacture products are always the same and always appear in the same sequence, but have variable process parameters depending on the specified product variant.
- Standard process sequence with variant: standard process sequence exists for manufacturing product variants of a product family. Some processes of the standard sequence can be added or neglected depending on the specification.
- General process framework: General process framework exists for all product variants, and the framework is filled with manufacturing processes from a predefined set. The sequence within the framework might vary.
- Variable process sequence: No predefined sequence of processes exists, and the manufacturing processes used might be totally new.

The process configuration can be applied for the "standard process sequence with variants" and the "general process framework", and is partially applicable for the "standard process sequence with variable parameters" (Wang, et al., 2015).

- **Definition 15: Process Configuration**

As we mentioned in the first Chapter, (Schierholt, 2001) concludes that the transference of the principles used in product configuration to the problem of process planning is called process configuration. So the essence of process configuration is the process planning (Wang, et al., 2015).

(Schierholt, 2001) defined process configuration as "the task of selecting the manufacturing processes needed to manufacture a product variant according to customer specifications from a fixed, predefined set of processes, putting these manufacturing processes into sequence, and generating production data for each process while respecting all compatibility constraints on how process may be combined and ordered".

Process configuration is more complex than product configuration because of the temporal constraints and resource constraints of production (Wang, et al., 2015). For example: some restrictions can be dynamics during the resolution process and also because the number of restrictions is unknown at the beginning (Wang, et al., 2015).

We can find some academics cases of production configuration of product families (Zhang, et al., 2012) (Zhang & Rodrigues, 2010) (Wu, et al., 2013). This production configuration concept was presented by (Zhang , 2007) based on a process platform as follows: "From an existing process platform in relation with a product family, the proper process elements, such as conceptual processes for individual product items (including the end product), operations, machines, tools, fixtures, cycle times, and setups are selected for new members of the associated product family, and subsequently arranged into routings, where in the process concepts for items are replaced with the detailed operations, for producing the given product; both the selection and arrangement of process elements are subject to constraints represented by configuration rules in the process platform".

(Wang, et al., 2015) applied the configuration technique to the process planning of product families. They introduced an algorithm in order to achieve the process configuration and developed a validation experiment on machining process configuration for a satellite plate panel. Formally speaking (Wang, et al., 2015) described the process configuration in the form of a tuple (P′, PM, RE, RU ) *where*:.

- P′ represents the sets of configurable objects in process, such as the set of activities, the set of constrained variables representing process characteristics and involved resources, the set of temporal constraints between activities, the set of resources constraints and the set of constraints on activity durations.
- PM defines the structure of process, represents the configurable process model.
- RE represents the process technical requirements of product.
- RU represents the rules and knowledge between configurable objects.

The result is either a process suitable for the target product or a no feasible solution. The feasible process in process configuration task could be called configuration scheme (Wang, et al., 2015).

Finally, (Aldanondo & Vareilles, 2008) introduced the Process Configuration view or routing configuration as already seen in section 1.1.2.2

– *"Given:*

*(i) a generic model of a configurable routing able to represent a family of production processes with all possible variants and options, that gathers:(1) a set of operations, (2) a set of resources with a required quantity, (3) a set of various constraints that restricts possible combinations of operations, resources and required quantities,*

*(ii) a set of inputs, where an input corresponds with a selection of an operation, a resource or a quantity value,*

– *Routing configuring can be defined as 'finding at least one set of operations with relevant sets of pairs (resource, quantity) that satisfies all the constraints and the inputs*."

### 2.1.3.2 Process Configuration Techniques

At the beginning when (Schierholt, 2001) presented the concept of Process Configuration, he described generic process structures using directed graphs. These structures called plan skeletons had the knowledge about sequences of processes.

In a same way as product configuration (section 2.2), authors interested in process configuration have shown that planning process could be also modelled and aided when considered as a CSP. For example, (Zhang & Rodrigues, 2010) studied the logic for configuring production processes using a dynamic modeling and visualization approach. (Zhang, et al., 2012) developed a constraint satisfaction approach for production configuration decisions taking into consideration the constraint identification, representation and evaluation. Moreover, the planning and scheduling problem was tackled by (Bartak , et al., 2010) using constraint satisfaction techniques.

Finally, (Wang, et al., 2015) applied the configuration technique to process planning of product families. They solved the process configuration task by generative constraint satisfaction problem (GCSP). The GCSP is an Extension of CSP that was proposed by (Stumptner , et al., 1998) (Stumptner & Haselböck, 1993).

### 2.1.4 Concurrent Product and Process Configuration (CPPC) Definition and CSP Model

In this section we present the Concurrent Product and Process definition and the related configuration techniques.

#### 2.1.4.1 Basic Definitions

- **Definition 16: Concurrent Product and Process Configuration**

From a product-line view point, a product consists of three parts (Dhungana, et al., 2017): "features representing the customer facing problem space, a BOM (Bill of Material) and BOP (Bill of Processes) representing the solution space". The result of product configuration is a manufacturing order consisting of a list of materials (BOM) and a set of production operations (BOP) acting on materials (Dhungana, et al., 2017). That is why it is inevitable to consider production constraints during the product configuration. The analysis of this kind of problem has to become concurring because the configuration task impacts the product domain and process domain at the same time (Aldanondo, et al., 2008). On one hand, product configuration decisions may have strong consequences on the planning of its production process. On the other hand, planning decisions can provide hard constraints to product configuration. Dealing with product and process configuration in an independent way can cause inconsistencies due to the consequences and constraints between both domains (Aldanondo, et al., 2008).

In the previous sections we have presented that (Aldanondo & Vareilles, 2008) summarized a Product Configuration definition and introduced the Process Configuration view or routing configuration. They analyzed the product domain and the process domain in a similar way, because associating the elements that characterize each definition we can find similarities between product and process environments, as shown in Table 1.

*Table 1- Similarities between Product and Process*

| Product | Process |
|---|---|
| ▪ A set of component groups. | ▪ A set of operations. |
| ▪ A set of product properties. | ▪ A set of resources with a required quantity. |
| ▪ A set of various constraints that restricts possible combinations of components and/or property values. | ▪ A set of various constraints that restricts possible combinations of operations, resources, and required quantities. |

| Product | Process |
|---|---|
| ▪ A set of customer requirements, where a requirement corresponds with a selection of a component or a property value. | ▪ A set of inputs, where an input corresponds with a selection of an operation, a resource or a quantity. |

As a result, there are elements of the process domain associated for each element of the product domain. For example:

- A set of component groups is related to the set of operations
- A set of product properties is specific related to the set of resources with a required quantity.
- A set of various constraints in the product domain is related to the set of various constraints in the process domain.
- A set of customer requirements linked with the product is related to the set of inputs for the process.

For that reason, (Aldanondo & Vareilles, 2008) showed how Process Configuration could be achieved with respect to Product Model. For example: the existence of an operation in a configured process routing can depend on the configured product (Aldanondo & Vareilles, 2008).

Based on this background we take back the definition of Concurrent Product and Process Configuration that we proposed in the first chapter "as the task of configuring a product and its related process at the same time, in order to meet technical and particular customer requirements"

#### 2.1.4.2 Concurrent Product and Process Configuration Techniques

Mass customization can encompass the management of the entire product cycle, from the customer's order to the final manufacturing (Aldanondo & Vareilles, 2008). Therefore, it is necessary to extend the configuration techniques to the process planning.

Some researchers incorporated product configuration with process planning (Pitiot, et al., 2014) (Campagna & Formisano, 2013). (Campagna & Formisano, 2013) described a modeling framework (PRODOC) that allows to model a product and its production process. They described the main features and capabilities offered to model production processes and to link them with the corresponding products models. They showed that processes could be modelled in terms of activities and temporal relations between them, considering resource production/consumption and interdependencies between process executions and product productions. (Aldanondo & Vareilles, 2008) extended the product configuration to downstream process configuration. They divided process configuration into routing configuration and operation configuration which could both be considered as CSP.

For handling the numerous constraints associated with product and process variety, (Zhang, et al., 2012) have developed a constraint satisfaction approach to facilitate production configuration. They formulated a domain-based model to conceptualize the production configuration process, involving inter-connections among multiple domains in conjunction with diverse domain decision variables and constraints. The production configuration was formulated as a constraint satisfaction problem (CSP), using a constraint heuristic for the search of the solution.

(Dhungana, et al., 2017) proposed to integrate product and production configuration by using new methodology for variability management in smart production ecosystems. They proposed novel concepts and algorithms for a holistic configuration approach required to support product designers, factory operators and end users in a common market place. They used the CSP paradigms for solving the combinatorial problem.

In (Aldanondo, et al., 2008) the product configuration and planning problems are considered concurrent as two Constraint Satisfaction Problems. The objective is to propagate decision consequences between the two domains by the use of constraints. They proposed to couple together interactive product configuration tools with process planning tools in order to pass decisions made from one to the other. In particular, they proposed to associate product configuration and production planning in order to allow: (i) the propagation of the consequences of each product configuration decision toward the planning of its production process; (ii) the propagation of the consequences of each process planning decision towards the product configuration. This should reduce or avoid planning impossibilities due to product configuration, and configuration impossibilities due to production planning.

Configurable products are modeled as a set of components (Aldanondo, et al., 2008). Each component is associated to a set of properties representing its configurable characteristics. Moreover, a set of constraints restricts possible combinations of components and property values. Production planning is addressed considering a production process as a set of task entities. A task entity is defined with: temporal parameters (start time, finishing time, and duration), resource parameters (required resource, and quantity of required resource), compatibility constraints (linking duration with required resource and/or required resource quantity) (Aldanondo, et al., 2008).

Coupling constraints are used to link a product with a process model (Aldanondo, et al., 2008). A coupling constraint is a compatibility constraint that links a variable of the configuration model with a variable of the planning model. Any variable of the configuration model can belong to a coupling constraint. On the planning model side, resource parameters and duration variables can be involved in coupling constraints. A resource parameter in a coupling constraint allows to propagate the impact of a configuration decision on the selection of the required resource and/or resource quantity (reverse behavior from resource selection to product configuration is also possible). A temporal parameter duration in a coupling constraint allows to propagate the impact of a configuration decision on the modulation of the duration of a task (reverse behavior from duration modulation to product configuration is also possible).

(Aldanondo, et al., 2008) showed that it is possible to manage interactions between product configuration and production planning. However, it´s only a primary result on the study of coupling process with product configuration.

#### 2.1.5 Synthesis

The analysis and modeling of both product and process domains has to be done simultaneously because of the concurrent characteristic, the multiple interactions and inter-relations that exist between the two domains. The configuration of each of these two domains can, without any doubt, be considered as a constraint satisfaction problem. Consequently, we consider the CSP framework for following modeling issues. In terms of kinds of CSP, we consider discrete variables and constraints for the configuration aspect and continuous variables and constraint for the criteria evaluation. As we are going to optimize the problem with evolutionary approaches, we will consider CSP with a static structure (no addition of variables and constraints in the problem during solving) in order to avoid the need to adapt evolutionary algorithm to chromosome with a structure that changes during optimization.

## 2.2 Optimization of CPPC: definitions, OCSP Model and multiobjective issues

Next, we summarized the most important definitions related to the problem of the Optimization of the Concurrent Product and Process Configuration (O-CPPC), the applicable optimization methods, the proposed method (CFB-EA) and related works.

### 2.2.1 Basic Definitions

- **Definition 17: Constrained Optimization**

Constrained optimization is the process of optimizing an objective function with respect to some variables in the presence of constraints on those variables. These constraints are expressed as a set of relationships that the variables have to satisfy. The constraints are usually presented as equalities, inequalities or compatibility tables.

- **Definition 18: Constraint satisfaction and optimization problem (CSOP)**

Constrained Satisfaction and Optimization Problems (CSOP) extend CSP with optimization needs. Solving a CSP means finding a feasible solution while solving a CSOP means finding a feasible and optimized solution (Eiben, 2001). Formally speaking CSOP is defined by (Tsang, 1993) as a quadruplet (X, D, C, f), gathering a CSP with an optimization function "f' which maps every solution tuple to a numerical value:

- *(X, D, C) is a CSP, and if S is the set of solution tuples of (X, D, C),*
- *Then $f(S) \rightarrow$ numerical value.*
- *Given a solution tuple T, we call f(T) the f-value of T.*

The task in a CSOP is to find the solution tuple with the optimal (minimal or maximal) f-value with regard to the application dependent optimization function f (Tsang, 1993).

- **Definition 19: Combinatorial Optimization**

In optimization field, combinatorial optimization consists of finding an optimal solution from a finite set of feasible solutions (Gupta & McGovern, 2011).

Usually the set of possible solutions is very large so much so that exhaustive search is not tractable. In the CPPC problem, it corresponds to the fact that decision variable domains (D) are discrete. Because the space of possible solutions is typically too large, the solving methods are generally suboptimal and include heuristics and metaheuristics.

**Definition 20: Constraint satisfaction and multiobjective optimization problem (CSMOP)**

This formalism extends the CSOP case with an objective function with several criteria, often contradictory. The fitness function of CSOP is therefore $f(S)=f_1(S), f_2(S),\ldots, f_t(S)$:

- *$f_i(S)$ is the $i^{th}$ objective function,*
- *S is a solution that respects C constraints,*

- *t is the number of objectives.*

Solving such problem consists in finding not only one solution but the set of solutions that represents the best trade-off or “compromise surface” between objectives. To define formally this set, the notion of optimal Pareto Front based on Pareto-dominance concept, is often used.

- **Definition 21: Pareto-dominance**

For any two solutions $a$ and $b$, $a$ dominate $b$ if:

$$\forall i, f_i(a) \leq f_i(b) \ and \ \exists i, f_i(a) < f_i(b), i = 1, 2, \dots, t$$

- $f_i(a)$ is the $i^{\text{th}}$ objective function,
- $a$ and $b$ are solutions that respect $C$ constraints,
- $t$ is the number of objectives.

- **Definition 22: Pareto optimality**

A solution $a$ is called Pareto optimal, if there is no other solution that dominates it.

The set of Pareto optimal solutions is called Pareto Front. Multiobjective optimization consists in finding this optimal set. But, as solution space is very large, the aim is to find the best approximation in a reasonable computation time.

### 2.2.2 Optimization and Configuration related works

The globalization forces many industries to change from mass production to mass customization, because it is necessary to respond to the customers’ requirements in a fast way with high quality and reasonable cost (Pine , 1993).

As mentioned by (Li, et al., 2006) and (Wei, et al., 2014), product configuration is one of the most important technology in the environment of mass customization. That´s why has been recognized as an effective means to implement it (Zhou, et al., 2008).

(Zhou, et al., 2008) and (Li, et al., 2006) agreed that most of the existing literature is mainly focuses on generating feasible configuration solutions from an engineering perspective using constraints-based and knowledge-based applications, which makes it very difficult to optimize design of product configuration (Li, et al., 2006). (Jiao & Zhang, 2005) explained that meeting customer’s individual requirements through product configuration is essentially an optimization problem. But the traditional product configuration optimization targets are mostly single (Wei, et al., 2014).

(Li, et al., 2006) emphasized that we can get a huge number of possible product configuration solutions from the product model despite a great variety of constraints. So they concluded that the goal of the configuration process is to find feasible solutions that satisfy customer requirements and product constraints. That´s why they presented an approach based on multiobjective genetic algorithm to solve this kind of problem. First, they discussed the configuration product model, then the multiobjective optimization problem of product configuration is described with its mathematical formulation. Finally, a multiobjective genetic algorithm is designed for finding the Pareto optimal for the problem.

(Du, et al., 2014) formulated a Stackelberg Game Theoretic Model for joint optimization of product family configuration and scaling design. In a bi-level decision structure reveals coupled decision making between module configuration and parameter scaling.

(Zhou, et al., 2008) proposed a new optimization approach for customer driven product configuration for assemble-to-order manufacturing enterprises. First they established a configuration space for targeting the diversity of customer needs and a utility function is employed to model and measure customer preference. They formulated the mathematic model that maximizes the ratio between overall utility and cost from the perspective of the customers and manufacturers. Finally, a genetic algorithm is adopted to solve the combinatory optimization problem where a nested encoding scheme and multiple constraints handling are incorporated to improve the performance of configuration solving.

(Wei, et al., 2014) mentioned that how to select the correct module to form the optimal product configuration scheme attracts increasing attention in the field of configuration optimization. So they presented and discussed the multiobjective optimization and evaluation method of modular product configuration design scheme. They proposed an approach based on genetic optimization algorithm and fuzzy-based select mechanism to solve the configuration optimization problem.

(Song & Chan, 2015) developed a study to optimize the configuration of product-extension service (PES). Because most of the configuration optimization models are product-related, and they are not suitable for PES configuration optimization. They proposed an optimization model for PES configuration. The model simultaneously takes service cost, service response time, and service performance as the optimization objectives. The model of service configuration is solved with the non-dominated sorting genetic algorithm II (NSGAII) to obtain the optimized service configuration set. The validation of the proposed model in elevator service configuration shows that it can be used as an effective method for PES configuration.

(Liu, et al., 2011) proposed that components in a product can be clustered into several modules according to some requirement. The authors defined a clustering issue as an optimization problem and identify its possible computation scale. Discrete particle swarm optimization (PSO) is applied to seek the optimal in the whole solution space, and it is proved as an effective method with an example of printer.

Finally, (Tang, et al., 2017)concluded that most of the configuration studies are focused on the cost or the customer utility, but ignore the environmental concern which becomes an important design criterion due to the rising awareness of environmental protection. They developed a new bi-objective optimization model integrating environmental concerns into product configuration. In their optimization model, the customer satisfaction Index (CSI) and the greenhouse gas (GHG) emissions of a product is formulated as optimization objectives. Moreover, to solve the established optimization model, they proposed a two-phase approach. In the first step, the relative weight of each function module is calculated and the candidate instance is determined by filtering the instances which do not satisfy the selection constraint. Then, the product configuration can be generated based on a multiobjective genetic algorithm. Finally, a numerical case study is introduced for testing the effectiveness of the proposed method.

### 2.2.3 Multiobjective decision aiding process

Several approaches could be used to solve a multiobjective optimization problem. The choice depends on the difficulty of the problem to solve (nature of evaluation functions, size and complexity of the problem in terms of number of variables, size and nature of their domains, constraints density, etc.) and also on desired decision support process.

Exact mathematical approaches like branch and bound have a limited field according to the complexity of the problem. Metaheuristics are then wildly used for complex problems. They don’t guarantee the optimality but give near-optimal solutions in reasonable computation time.

Considering decision support in multiobjective context, three main approaches could be considered: “a-priori”, interactive or “a posteriori” approaches.

- In “a-priori” approaches, user must prioritize or weight objectives. Optimization is thus transformed in a mono-objective problem. The major issue is thus to be able to prioritize objectives.

- In interactive approaches, user guide optimization towards a solution satisfying the selected compromise during optimization. This kind of approach does not guarantee to explore all areas of the search space but allows to explore faster an area of interest of the user.
- In “a posteriori” approaches, the multiobjective aspect is preserved during optimization step and algorithm must find a complete set of solutions that represent the best compromise between objectives (Pareto Front). The trade-off between objectives is done by the user after the optimization when he selects a solution among the given set.

This “a posteriori” way clearly complicates the optimization and requires an additional selection step for the user. But it avoids “blind” choice between objectives. The user can see different possibilities and decide his/her own specific compromise. Our approach belongs to this kind of decision aiding process. Nevertheless, improvement proposed in chapter 5 is inspired by interactive possibilities.

Even in a-posteriori approaches, there is two ways to deal with a multiobjective optimization: aggregation (or scalarization) approaches or Pareto-based approaches. Scalarization methods decompose objective space and try to find best solutions in each area. Pareto-based approaches maintain various objectives separate and use Pareto-dominance concept to evaluate solutions.

Various metaheuristics like particle swarm or evolutionary algorithm (EA) could be used in this optimization task. EAs show a great interest for multiobjective optimization due to their population-based and easy to implement skills.

Evolutionary algorithms deal simultaneously with a set of possible solutions (the so-called population). This allows to find several members of the Pareto optimal set in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the traditional mathematical programming techniques. Additionally, evolutionary algorithms are less susceptible to the shape or continuity of the Pareto front (e.g., they can easily deal with discontinuous or concave Pareto fronts), whereas these two issues are a real concern for mathematical programming techniques.

### 2.2.4 Presentation of EA

Evolutionary algorithms are inspired by the biological evolution of species and appeared at the end of the 1950´s. They are part of a branch of artificial intelligence concerning optimization algorithms called metaheuristics. These are generic optimization algorithms (applicable to a wide family of combinatorial and multiobjective problems). Evolutionary Algorithms is the name of the family of methods using evolutionary concepts. They gather various subgroups like evolution strategies (Rechenberg, 1965), evolutionary programming (Fogel , et al., 1966) or genetic algorithms (Holland , 1992).

Formally speaking, a generic EA consists of four operations, including reproduction, mutation, recombination, and selection, and all these operations are repeated until the algorithm converges to a certain point with some criteria satisfied (Dubitzky, et al., 2013). In an EA, each candidate solution is represented as a chromosome. In each step of EA, the chromosomes compete against each other and those representing poor solutions will be kicked out before next step. To evaluate the chromosomes, a fitness function is defined as the objective function.

(Zitzler, et al., 2002) summarized that after the first studies on evolutionary multiobjective optimization (EMO), a number of Pareto-based techniques were proposed like MOGA (Fonseca & Flemming, 1993) or NSGA (Srinivas & Deb, 1994). They demonstrate the capability of EMO algorithms to approximate the set of optimal trade-offs in a single optimization run. Then a couple of elitist multiobjective evolutionary algorithms were presented at this time. The two more known are SPEA2 (Zitzler, et al., 2002) and NSGA-II (Deb, et al., 2002). More recently, new approaches have proposed using scalarization technics like ε-MOEA (Deb, et al., 2003) or MOEA/D (Zhang & Li, 2007). They convert a multiobjective problem in a collection of single-objective problems. Solving each sub-problem gives a point in Pareto front. Those scalarization methods show a great interest for many-objective problem (more than 4 objectives) (Wagner, et al., 2007).

In this research, we will use and improve an adapted version existing of the Evolutionary Algorithm SPEA2 because it can provide solutions on a Pareto front in a first efficient way for this kind of global optimization problems with multiple objectives. We chose it because it is widely used and recognized to solve difficult, combinatorial and multiobjective optimization problems and it’s easy to implement. The concepts discussed could also be adapted for other similar metaheuristics (particle swarms, ant colonies,...) or other EA variants.

The main ideas of SPEA2 are a) the evaluation of a solution takes Pareto-dominance and the local density of solutions into account, b) a set of the best and most diversified solutions are preserved in a separate archive, c) a binary tournament in the archive is used to select parents for a crossover. The SPEA2 algorithm has been adapted to take into account constraints of the problem using a filtering engine. It leads to CFB-EA proposed in (Pitiot, et al., 2013).

### 2.2.5 Constraints handling in EA

EAs were initially proposed for very large and unconstrained solution spaces. They have been adapted by many authors in order to handle constraints. On his website, (Coello, s.f.) maintains a "List of References on Constraint-Handling Techniques used with Evolutionary Algorithms" that can be organized according to the following six kinds of approaches.

- Penalty function. The idea is to reduce or penalize fitness value according to constraint violation. Thus, solutions that do not respect constraints are discarded.
- Stochastic ranking. The idea is to modulate previous over/under penalization of the penalty function with a ranking process.
- Epsilon constrained method. The idea is first to minimize the number of violated constraints, then optimize the objective function.
- Multiobjective. The idea is to consider constraint violation as a single objective and to associate it with the original objective function in a multi-criteria problem.
- Feasibility rules. The idea is to compare all pairs of solutions, in a binary tournament, with three rules mixing fitness value and constraint violation level.
- Special operators. The idea is to deal only with feasible solutions through repairing methods or preservation of feasibility methods.

The CFB-EA algorithm belongs to the last kind of approach, which only allows feasible solutions in the archive. CFB-EA complete SPEA2 with the addition, during: (i) the initial population generation process, (ii) the crossover and mutation process, of some constraint filtering that prunes search space and prevents inconsistent solutions in the population. Six parameters are required: size of archive, size of population, number of generations or any stopping criterion, crossover probability for individual selection, mutation probabilities for individual and gene selections. The CFB-EA algorithm is illustrated on figure 8.

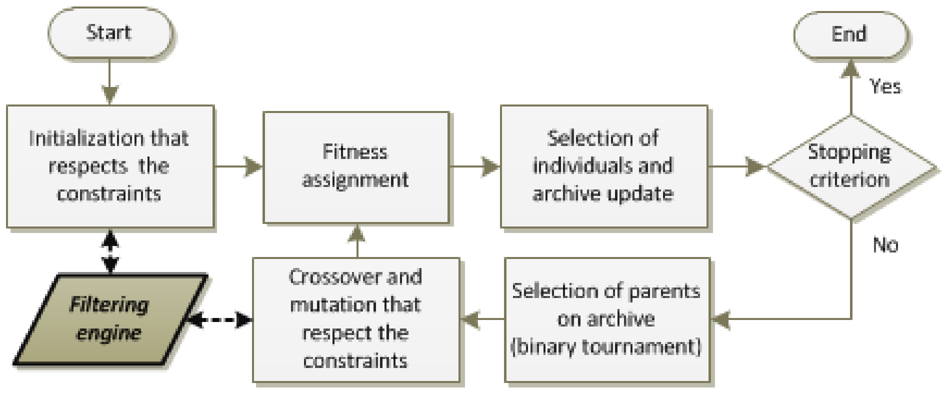

Start
End
Yes
Initialization that respects the constraints
Fitness assignment
Selection of individuals and archive update
Stopping criterion
No
Filtering engine
Crossover and mutation that respect the constraints
Selection of parents on archive (binary tournament)

*Figure 8- CFB-EA algorithm (Pitiot, et al., 2013)*

### 2.2.6 Synthesis

Given the following specificities of the configuration problem: (i) the problem size, which can vary greatly, (ii) the size of the problem to optimize, which depends on the amount of elementary requirements to be processed before optimization, (iii) the constraints level which is rather low (the goal of companies is to sell products and so many solutions are possible), (iv) multi-criteria optimization is most often necessary; even if some works have investigated configuration optimization using case-based reasoning, most of the published material considers metaheuristics. Due to their population-based search, their multi-objective aspect and their genericity, metaheuristics like particle swarm optimization (PSO) (Yadav, et al., 2012) or Evolutionary Algorithms (EA) (Wei, et al., 2014) (Dou, et al., 2016) (Tang, et al., 2017) are logically suitable for configuration optimization. They have to integrate constraints handling to manage unfeasibility in their optimization process. We position our contribution in this work stream and will use the Evolutionary algorithms CFB-EA (Pitiot, et al., 2013) which is based on SPEA2[6]. Conclusion and confirmation of thesis proposition

It´s inevitable to consider process configuration during the product configuration. The analysis of this kind of problem has to become concurring because there are multiple interactions and interrelated variables between both domains. It has been shown that the Constraint Satisfaction Problem framework can be efficiently used to model product and process configuration and its optimization.

, For the Optimization of the Concurrent Product and Process Configuration problem (O-CPPC), we have shown the interests of metaheuristics and more specifically evolutionary approaches. That´s why we have chosen CFB-EA which is an existing adapted version of the Evolutionary Algorithm SPEA2 that is widely used and recognized to solve these kinds of difficult problems.

There is no standard to analyze the O-CPPC, and the existing works dealing with the sequential association of Optimization plus Concurrent Product and Process Configuration are rather theoretical with evaluations that consider most of the time a single problem without a detailed design of experiments. Consequently, we confirm the initial goals of the thesis and next chapters contain that will deal with: (i) generic modeling propositions, (ii) evaluations of CFB-EA optimization approach and finally (iii) CFB-EA optimization improvements.

[6] SPEA, an acronym for Strength Pareto Evolutionary Algorithm

# 3. Generic model of O-CPPC

In order to address our first research question "Is it possible to propose a generic model of the CPPC problem that can avoid case dependency when evaluating and comparing optimization methods?" the goal of this chapter is to define and discuss a generic model that allows to generate benchmark for O-CPPC optimization problems. A benchmark is a set of model instances representative of a specific optimization problem and which allows testing of optimization algorithms and validation of their accuracy for the addressed problem. We need to generate various instances of the O-CPPC problem that represent the diversity and the complexity of industrial cases. In the following sections, we will define the product configuration generic model, process configuration generic model and their coupling. This will enable us to identify the O-CPPC key characteristics.

## 3.1 Product configuration generic model for benchmark

### 3.1.1 Product as a set of physical/functional modules

In the design community many works have proposed to characterize or represent the product development. Axiomatic Design proposed by (Suh, 2001), (Suh, 1990), Design Structure Matriz (Stewart, 1981) and Function Behavior Structured (Gero, 1990) propose different domains or views of the product as: functions, requirements, behavior, physical components, process and resourc. (Lindemann, 2007) proposes a mapping of the four domains: functions, components, process and resources with a DSM approach. Furthermore, all these authors insist on decomposition aspects with various criteria: functional, physical or temporal.

For our product configuration modeling problem, the identification or representation of physical components is an essential requirement because it allows us to deal with the bill of material concept which is a key point of configuration. According to configuration definitions of chapter 2, these components are gathered in component families that are considered as a product configuration variables. Chapter 2 also explains that the product, its sub-assemblies and its components can also be characterized with descriptive attributes with some kind of descriptive or functional views. We therefore consider in our model that the product, with its sub-assemblies and components is described by a set of configuration variables which are decision variables that are either families of components (noted foc) or functional descriptive variables (noted fdv), the model is therefore: $\{\{fdv_i\},\{foc_j\}\}$. When the definition domain of all these variables is reduced to a singleton, the configuration of the product with all sub-assemblies and components is over. As we deal with configuration, product configuration constraints limit the possible combinations of previous variables values. Configuration constraint (noted $cst_k$ ) can be between $fdv_i$ or between $foc_j$ or between both $fdv_i$ and $foc_j$. For a car example, constraints can be: between $fdv_i$ (speed, gas consumption), between $foc_j$ (Engine_ref, Tire_ref), or between both $fdv_i$ and $foc_j$ (speed, Engine_ref).

As recalled at the beginning of this section, a product can be decomposed according to different criteria. Axiomatic design (Suh, 2001) and design structure matrix (DSM) (Stewart, 1981) allow to identify functions associated with sub-assembles that fit very well the modular need of the configuration problem. We follow this idea and consider that the product is a set of physical-functional modules. For example, if the configured product is a car, physical-functional modules can be: engine, body, electrical system, transmissions, audio system, etc. Consequently, each module is a set of families of components and functional descriptive variables, the previous configuration constraints are either between modules ($cst_{kb}$) or inside a module ($cst_{ki}$) and the generic configuration model becomes:

$\{ \{ \{fdv_i\},\{foc_j\},\{cst_{ki}\} \}, \{cst_{kb}\} \}$.

As our goal is to optimize the CPPC, we propose to add criteria variables to each product module and we make the assumption that an aggregation formula allows to deduce each criterion value for the whole product. In our case for the product we only consider a price criterion. Cost and selling price are two different ways to economically evaluate a product. Costs are what pays the enterprise (raw material, components or resources) while selling price is what customer pays. Coming from value analysis domain, the functional descriptive variables are related with selling price while components and resource are related with the product cost and the process cost. As configuration takes place during negotiation with customer, internal costs have to be hidden and they will be changed in selling price (commercial strategy is included like margins or discounts). Consequently, we consider that the price is modeled with selling price variables (noted $sp_m$) obtained thanks to a formula involving configuration variables that can be both families of components and/or functional descriptive variables. This price formula is modeled with numerical function constraints (notes $cst_{sp}$) according to configuration variables that are either inside each module ($cst_{spi}$) or at the product level for cost aggregation ($cst_{spp}$). The generic configuration model becomes:

$\{ \{ \{fdv_i\},\{foc_j\},\{cst_{ki}\},\{sp_m\},\{cst_{spi}\} \}, \{cst_{kb}\}, sp,\{cst_{spp}\} \}$.

An example of a resulting product model is shown in figure 9. This example gathers two modules. The first one, “module 1”, is very simple:

- It gathers three families of components: $foc_{1\text{-}1}$, $foc_{1\text{-}2}$ and $foc_{1\text{-}3}$, there is no descriptive attributes
- Two constraints ($foc_{1\text{-}1}$, $foc_{1\text{-}2}$) and ($foc_{1\text{-}2}$, $foc_{1\text{-}3}$) that shows component compatibilities,
- Each component family is linked to a criteria variable which is the selling price in this case ($sp_{1\text{-}1}$, $sp_{1\text{-}2}$, $sp_{1\text{-}3}$).

The other module, “module 2”, is more complex:

- In the upper part, a descriptive variable ($fdv_{2\text{-}1}$) allows to identify component ($foc_{2\text{-}1}$) this component allows to quantify a selling price variable ($sp_{2\text{-}1}$).
- In the lower part, a combination of two components ($foc_{2\text{-}31}$, $foc_{2\text{-}32}$) allows to quantify a selling price variable ($sp_{2\text{-}3}$).
- In the middle part, a combination of two descriptive variables ($fdv_{2\text{-}21}$, $fdv_{2\text{-}22}$) allows to identify a component ($foc_{2\text{-}2}$) that allows to quantify a selling price variable ($sp_{2\text{-}2}$).
- Two compatibility constraints between descriptive variables ($fdv_{2\text{-}1}$, $fdv_{2\text{-}21}$) and between component families ($foc_{2\text{-}2}$, $foc_{2\text{-}.31}$) are also present. An inter-module constraint between the two modules exists between components families of each module ($foc_{2\text{-}32}$, $foc_{1\text{-}.2}$).
- Finally, all selling price variables are linked with a numerical constraint (a sum in most of the cases) in order to get the selling price of the whole product.

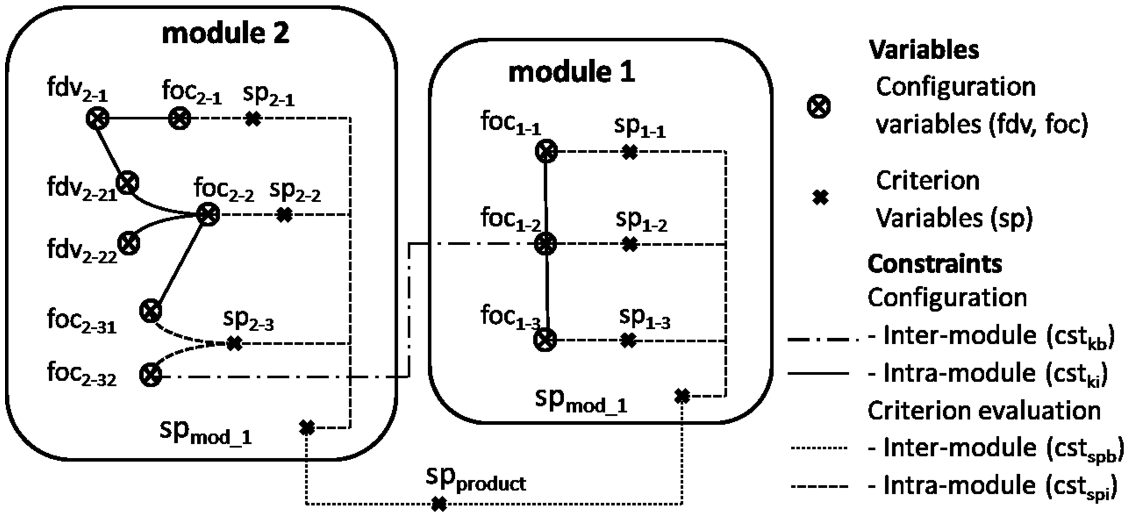


*Figure 9- First CPPC product generic model for benchmark*

### 3.1.2 Module as a set of Product Configuration Evaluation Patterns

Given previous generic modeling propositions and in order to generate examples of models, assuming the definitions of:

- a number of modules,
- for each module: a number of component families, descriptive variables and criterion variables,
- for each component family or descriptive variables: a number of possible values,
- for each criterion variable: a possible range of values,

a solution could be to randomly generate a set of constraints that reduces more or less the quantity of possible combinations of all previous variables. This way to process, given the randomly generation of constraints, generates descriptions that don't have sense or that don't correspond to any company configuration situation. For example:

- A descriptive variable that is not linked to a family of component or a selling price variable with a constraint has no sense; this would mean that a product characteristic (as for example power, length, color) has no impact on any component selection or on module price.
- Similarly, a family of components that is not linked to a selling price variable would mean that a selected component (as for example an Engine-Ref) has no impact on a module price.

Consequently, typical industrial configuration situations have been analyzed. We observed that in the majority of product configuration models, some small groups of variables are strongly connected and that some common generic sets of variables and constraints are frequently repeated. We therefore propose to identify a small number of these generic sets that we call Product Configuration/Evaluation Pattern (noted PCEP) and to consider that a module is a set of such patterns. Each PCEP gathers a set of decision configuration variables (foc and/or fdv), at least one criterion variable (selling price, noted sp, in our case) linked by configuration and evaluation constraints. In the following we consider just one criterion for clarity; of course other criterion could be added.

Four Product Configuration/Evaluation Patterns (PCEP) have been identified and are now presented and discussed. They are shown in figure 10.

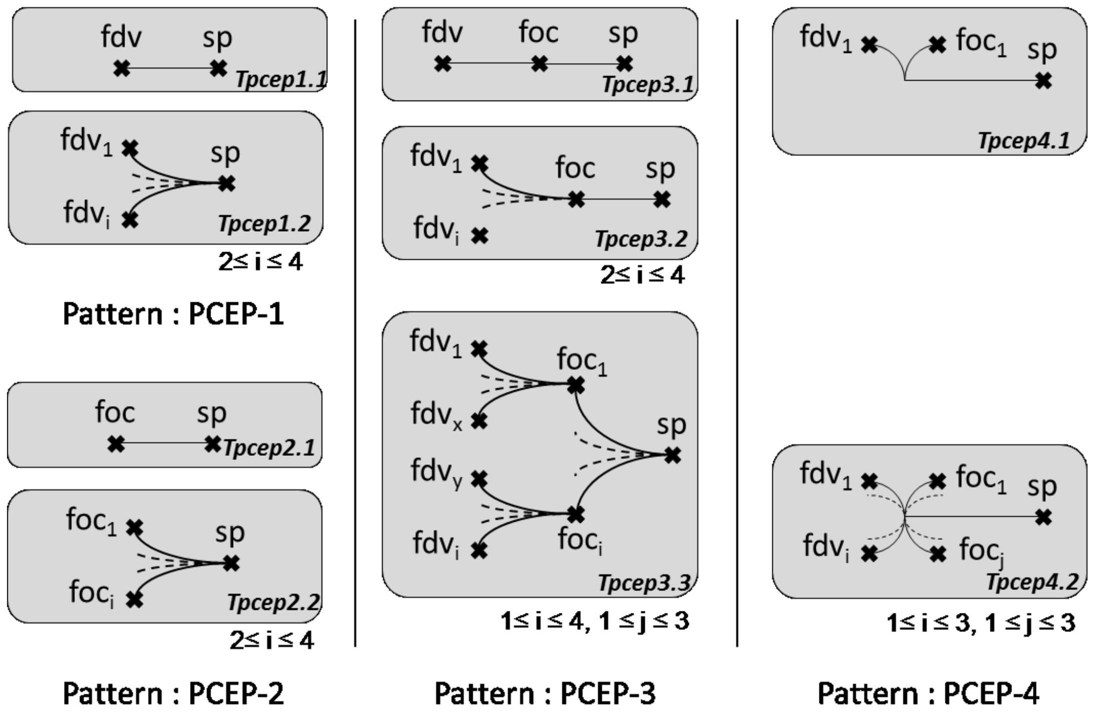


*Figure 10- Four Product Configuration/Evaluation Patterns (PCEP)*

3.1.2.1 Patterns PCEP-1 and PCEP-2: single evaluation pattern

These two patterns gather a configuration and an evaluation constraint. They correspond to a single point of view analysis, either functional (PCEP-1) or physical (PECEP-2). They express the fact that a set of compatible components or functions is linked to a specific criterion or selling price variable.

PCEP-1.1 and PCEP2.1 just show that one product descriptive variable or one family of components influences one price variable. This can correspond with a kind of very simple component price catalogue. For example, for PCEP-1.1, a specific cable is always present in a module, only the length or the cable can be selected during configuration and the selling price varies only according with this length variable.

PCEP-1.2 and PCEP2.2 show that a criterion variable depends of more than one descriptive variable or component family. In fact, this constraint can mix two constraints, one that shows allowed combinations of descriptive variables or component families ($cst_{ki}$) and one that provide the criterion or price of each association ($cst_{spi}$). For example, for PCEP-1.2, the price of a machine window depends of its height and width, but this constraint can also specify that extreme height and width are incompatible. Similarly, for PCEP-2.2, the price of the association of two components engine and gear-box depends of the two selected components but for power reason all gearbox is not compatible with all engines.

#### 3.1.2.2 Patterns PCEP-3: composed evaluation pattern

This pattern reflects some kind of a conventional design process: describe what you want with descriptive variables, choose technical solutions or components, and quantify a pattern criterion value that will be used later to deduce a module criterion.

PCEP 3.1 is the simplest, one descriptive variable identifies one component that quantifies one criterion or price. This is typically a simple catalogue, as for example a power need enable to identify an engine reference with a price.

PECP 3.2 shows that more than one descriptive variable can be necessary to identify a component. Again, a catalogue example for a machine window where four attributes (length, width, glass material and color) are necessary to identify a component with its price.

PECP 3.3 is something which is close to what we call a module because it can gather many descriptive variables and component families. However, here the idea is still to follow the process: each component is chosen with respect to different attributes, all the components families are combined to provide a criterion or price. The only difference with a module is that constraints between descriptive attributes associated with different families of component are forbidden. For example, the previous engine gear-box association can illustrate this pattern: (i) the engine component is chosen with respect to the three descriptive attributes max power, max torque and color, (ii) the gear-box is chosen with respect to the three descriptive attributes number of stages, admissible power and color, (iii) their association provides a price, (iv) it is not possible to express a constraint saying that they should both have a same color.

#### 3.1.2.3 Patterns PCEP-4: mixed evaluation pattern

For these patterns the previous process that considers in a kind of sequence descriptive attributes, component family and finally criterion is not present anymore. The criterion or price is directly a combination of both descriptive attributes and component families. For example, a kitchen worktop is a parametric component with a price than can be defined by a component reference of the worktop (aggregating material, thickness and finish) and the two descriptive variables length and depth.

#### 3.1.2.4 Module as a set of PCEP patterns

As a consequence, a module is a set of PCEP patterns and constraints can of course exist between two patterns inside a same module. The generic configuration model for benchmark becomes therefore:

- Product = (i) set of modules and configuration constraints between modules, (ii) product criterion variables and evaluation constraint that aggregates module criterion variables,
- Modules = (i) set of instance of PECP patterns and configuration constraints between pattern inside a module, (ii) module criterion variables and evaluation constraint that aggregates pattern criterion variables,
- Pattern = (i) set of variables: descriptive attributes, component families and criterion variables, (ii) set of constraints: configuration constraints and evaluation constraints

The model example provided in figure 9 is now updated in figure 11 with this notion of Product Configuration/Evaluation Patterns.

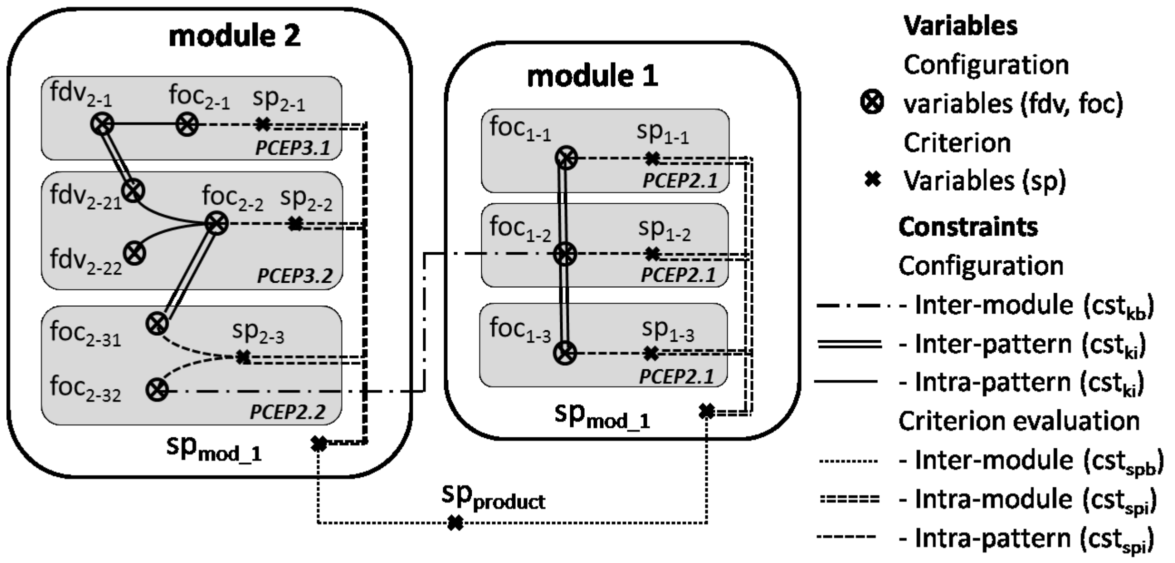


*Figure 11- Example of CPPC product generic model for benchmark*

## 3.1.3 Constraints patterns

Assuming now that all variables and constraints of the product generic model are described, it is necessary now to define the allowed combinations of each constraint. As in the beginning of the previous section we could think of a random generation of allowed combinations. But in order to be more representative or closer to the reality of company situations we also proposed constraint patterns and will first detail configuration constraints and then evaluation constraints.

A constraint could be an equation or a compatibility table according to the nature of variables involved (continuous or discrete). A compatibility table shows allowed and/or forbidden combinations of values of involved variables. In our generic model, configuration constraints are exclusively discrete and represented by tables of compatibility that link configuration variables; while evaluation constraints can use equations and tables of compatibility.

### 3.1.3.1 Product configuration constraints patterns

In our generic product model, a product configuration constraint is defined by: a type depending on the location of the linked configuration variables, a configuration constraint pattern and a constraint density. We define in next sections those concepts.

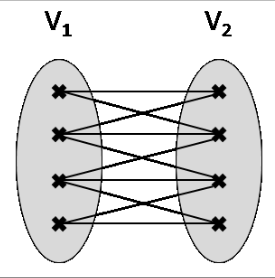
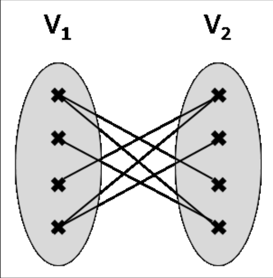
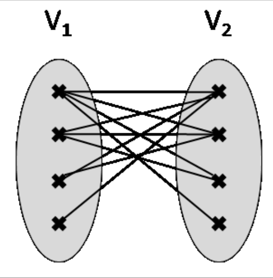
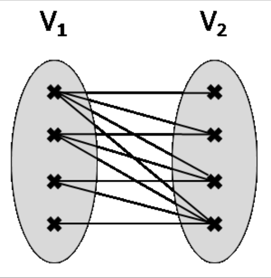
Type of product configuration constraints

As seen in previous subsection, product configuration constraints take place in different location of the model. We recall these types:

- Intra-PCEP, they link inside a PCEP either: (i) only descriptive variables, (ii) only families of component or (iii) both,
- Inter-PCEP, we assume that these configuration constraints link inside a module: (i) only descriptive variables, (ii) only families of component of different PCEP. This is done in order to keep links between functional and physical aspects at the PCEP level. For a realistic model, we limit the arity of those constraints to a maximum of four.
- Inter-Module constraints, we assume the same kind of restriction between modules and allow links between: (i) only descriptive variables, (ii) only families of component of different modules. Similarly, the maximum arity is set to four.

Configuration constraint patterns

In this sub-section, we therefore only consider constraints linking configuration variables as descriptive variables and/or families of component. The values of each of these variables can be either ordered or not. For example, there is, most of the time, no order when dealing with configuration variables as: color, shape, style or more generally with a variable under purely subjective selection. For the constraints taking into account this kind of variables, there will be no configuration pattern. Variables with values that are ordered can be for example: power, length, size, complexity, pressure more generally any variables with values that can be compared or ordered. These last examples are descriptive variables, but it is also very easy to order families of component with respect to their evaluation criteria. We have identified four various configuration shapes or configuration constraint patterns (named Tcp) that are illustrated with examples in figure 12. These examples show configuration constraints between two variables but they could be extended to three or four variables. For each of these variables, the values have been replaced by their order in order to be able to compare them with computation. Compatibility tables are illustrating by compatibility matrix. A cross in matrix corresponds to a compatible couple of values. Each value of each involved variable has to appear in at least one allowed tuple. (i.e. we have at least one cross in each line and each column in associated matrix).

| Similarity/Close level Pattern (Tcp1) | Dissimilarity Pattern (Tcp2) | Limit Pattern (Tcp3) | Comparing Pattern (Tcp4) |
|---|---|---|---|
| $\lvert V_i - V_j \rvert \le diff\_level$ | $\lvert V_i - V_j \rvert \ge diff\_level$ | $\sum_{i=0}^{j} V_i \ge limit$ | $V_i \le V_j$ |
|  |  |  |  |
| Ex : $\lvert V_1 - V_2 \rvert < 2$ | Ex : $\lvert V_1 - V_2 \rvert \ge 2$ | Ex : $V_1 + V_2 \le 5$ | Ex : $V_1 \le V_2$ |

Tcp1:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | X | X | | |
| 2 | X | X | X | |
| 3 | | X | X | X |
| 4 | | | X | X |

Tcp2:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | X | X |
| 2 | | | | X |
| 3 | X | | | |
| 4 | X | X | | |

Tcp3:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | X | X | X | X |
| 2 | X | X | X | |
| 3 | X | X | | |
| 4 | X | | | |

Tcp4:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | X | | | |
| 2 | X | X | | |
| 3 | X | X | X | |
| 4 | X | X | X | X |

*Figure 12- Four kinds of configuration constraints patterns*

- Tcp1: Similarity or close level pattern, this pattern corresponds to a similarity between various levels of decision variables. The parameter that controls the strength of the constraint is the allowed order difference between two values, in the proposed example the allowed difference must be less than 2. This can represent a design rule that maintains consistency between choices on a product. For example, this could lead to forbid the association of a luxury with low-cost components or to configure a house window with two extreme parameters: height (2 meters) and width (10 cms).
- Tcp2: Dissimilarity pattern, this pattern is exactly the opposite of the previous with a similar minimum of allowed order difference between two values. In the example of figure 12, we have shown the complementary combinatory of the previous pattern. This can represent a conception or organizational rule that forbids the selection of same level for selected variables. For example, this could lead to forbid the selection of similar dangerous material for various component or similar suppliers for different operations.

- Tcp3: Limit pattern, this pattern corresponds to some kind of a cumulative limit of various value order or levels. The orders of the configuration variables are cumulated with a sum or a product calculation (sum in the example of figure 12) and the cumuli should respect a maximum threshold which is the parameter of this pattern. It can represent a physical limitation. For example, a car autonomy (kms with a full tank) and average cruising speed (kms/ hour) follows this kind of pattern.
- Tcp4: Comparing pattern, this pattern corresponds to an inclusion of different orders in a unique way that can be explained as "a higher order can fulfill any lower order". In the example of figure 12, given an order of V1 all orders of V2 that are larger or equal are compatible. It can represent for example that a given power requirement can be fulfilled by an engine matching exactly this power need but also by all engines that have a greater power.

Configuration constraint density

Constraint density corresponds to the ratio of the number of tuples forbidden by the constraint divided by the number of possible tuples without taking the constraint into account (or Cartesian product of definition domain size of constrained variables). On the previous figure 12, it corresponds to the ratio of number of white boxes divided by the matrix total size in the compatibility matrixes. The constraint density is directly correlated with the strength parameter of previous configuration constraint patterns. It is important to note that a high constraint density allows a lower number of configuration solutions than a low constraint density. Therefore, higher the constraint density is, higher constrained is the problem.

#### 3.1.3.2 Evaluation constraint patterns

As for configuration constraints, evaluation constraints are characterized with a type, a pattern and a constraint density.

Type of product evaluation constraints

As seen in previous subsection, product configuration constraints take place in different location of the model. We recall these types:

- Intra-PCEP, they link: (i) descriptive variables and/or families of component with (ii) an evaluation variable inside a PCEP. Given the fact that configuration variables are discrete and evaluation variables are either integer or floats, these constraints can be discrete or mixed and defined thanks to table. In the case of a selling price criterion, these constraints correspond very frequently with some kinds of catalogues with prices.

- Inter-PCEP, they allow to aggregate evaluation variables of all PCEP of a single module into an evaluation variable for the whole module. In the case of a selling price, these constraints are most of the time just a simple sum calculation.
- Inter-Module constraints, they allow to aggregate evaluation variables of all modules of the configurable product into an evaluation variable for the whole product. Here again, in the case of a selling price, these constraints are most of the time just a simple sum calculation.

Evaluation constraint pattern

Given the fact that Inter-PCEP and Inter-module evaluation constraint are more or less aggregation calculation, we propose evaluation patterns only for Intra-PCEP pattern. Similarly, with configuration constraint pattern, we assume that the descriptive variables and/or families of component involved in the evaluation can be ordered. Three evaluation patterns have been identified as shown in figure 13:

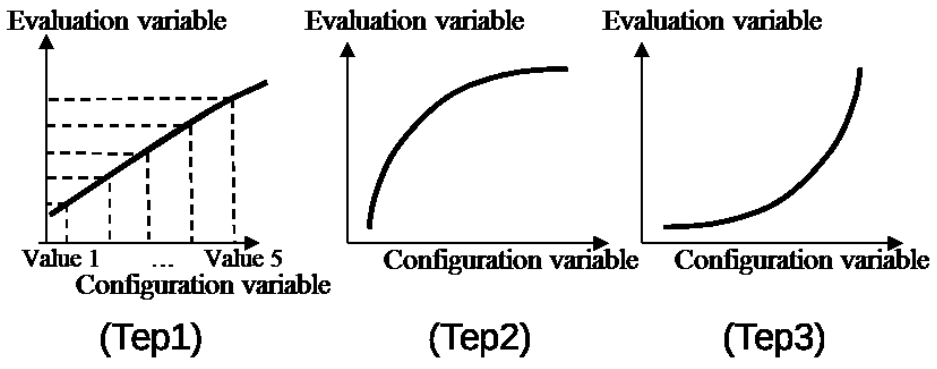


*Figure 13- Three kinds of evaluation constraints patterns*

- Tep1, Linear progression, in this case the criterion value increases proportionally with the order of configuration variables,
- Tep2, Decreasing progression, in this case with respect to configuration variables growth, the criterion value increase strongly at the beginning and less at the end,
- Tep3, Increasing progression, in this case with respect to configuration variables growth, the criterion value increase slowly at the beginning and much more at the end.

  When more than one configuration variable are considered in these patterns, a kind of average order is computed and associated with the pattern.

### 3.1.4 About Product architecture

We first recall some very basics about product architecture and show how our proposition fits the most frequent kind of architectures of configurable products.

### 3.1.4.1 Product architecture basics

(Ulrich, 1995) defined the architecture of a product as "the scheme by which the function of the product is allocated to physical components". More precisely the author defined product architecture as: (1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components. Another definition was presented by (Ulrich & Eppinger, 1995) as a "scheme by which the functional elements of the product are arranged (or assigned) into physical building blocks (chunks) and by which the blocks interact" as shown in figure 14. So the arrangement of functional elements into physical chunks becomes the building blocks for the product or family of products (Ulrich & Eppinger, 1995).

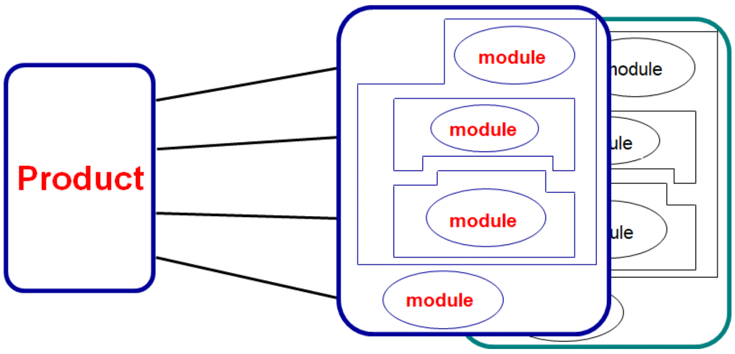


*Figure 14- Product Architecture Definition (Ulrich & Eppinger, 1995)*

Most authors agree on two main kinds of architectures: modular and integrated. Furthermore, a certain kind of modular architecture called platform is also considered in (Bonjour, 2008) (Bonjour, et al., 2009) (Bonjour & Micaëlli, 2010).

### 3.1.4.2 Modular architecture|

The first distinction in the typology is between a modular architecture and an integrated one. (Marti, 2007) characterized a modular system architecture by the property of near-decomposability, that is consisting of relatively autonomous subsystems. So in this architecture a module can be defined as "a special subsystem whose internal relationships are much stronger than the relationships with other subsystems" (Marti, 2007). Then (Ulrich, 1995) emphasizes that a modular architecture includes a one-to-one mapping from functional elements in the function structure to the physical components of the product and specifies decoupled interfaces between components. More specific, (Blackstone, 2013) explained that the modular architecture is a type of structure where the functional modules correspond to physical group of parts. The different physical pieces of parts have their own function, and there is an interaction between all modules (Blackstone, 2013).

Given these elements and the product generic model we propose, we can deduce that a modular architecture must show:

- A large number of constraints inside each product modules (Intra-PCEP and Inter PCEP) with a high constraint density meaning that the quantity of allowed combinations is small or the dependencies between descriptive variables and families of components are strong.
- A low number of constraints between modules (Inter-Module constraints) with a low constraint density meaning a high number of possible modules combinations.

In the example of figure 15, we have a three modules product where each module contains three PCEP patterns. The numbers of constraints are for Intra PCEP: 9, for Inter PCEP: 8 and for Inter modules: 3.

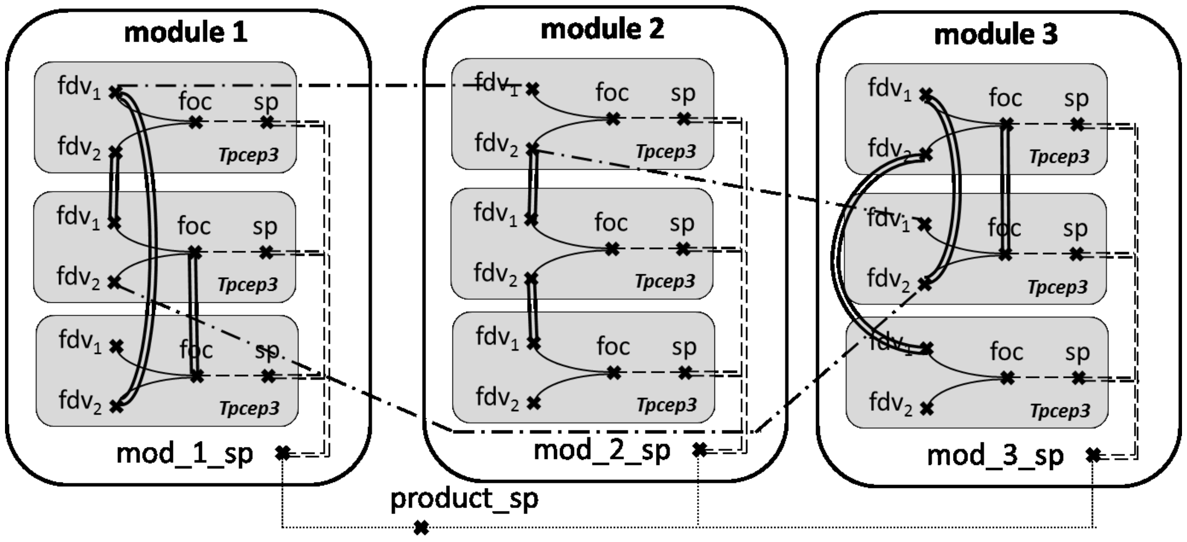


*Figure 15- Example of product model with a modular architecture*

3.1.4.3 Integrated architecture

(Ulrich, 1995) explained that an integrated architecture includes a complex (not one-to-one) mapping from functional elements to physical components and coupled interfaces between components. In this type of architecture, the modules are more dependent on each other and less easily distinguished (Marti, 2007).

Given these elements, we can deduce that an integrated architecture must show:

- A number of constraints inside each product modules (Intra-PCEP and Inter PCEP) which is closer to the number of constraints between modules (Inter-Module constraints)
- Constraint densities with closer values, meaning that the quantity of allowed combinations or the dependencies between descriptive variables and families of components is not affected by the fact that a constraint is Inter-module or Intra-module.

In the example of figure 16, we have a three modules product close to the one of figure 16. But for this one, the numbers of constraints are for Intra PCEP: 9, for Inter PCEP: 3 and for Inter modules: 7 and all their constraint densities are close.

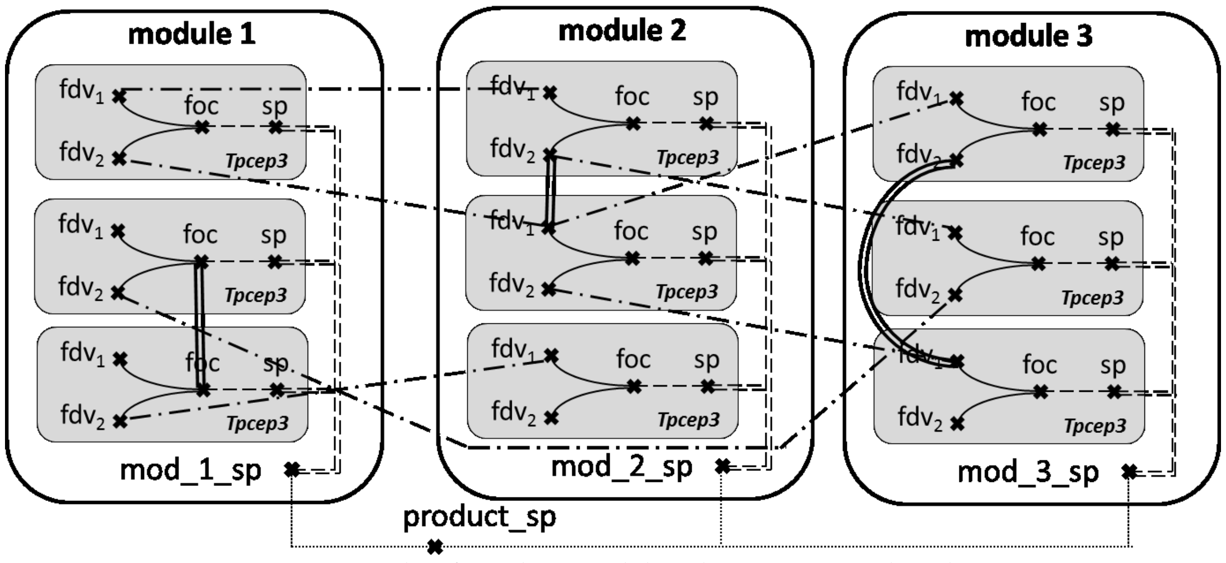


*Figure 16- Example of product model with an integrated architecture*

3.1.4.4 Platform architecture

An increasingly popular method to reduce complexity in configurable products is the product platform architecture. (Marti, 2007) explained that the product platform is a special case of product modularization; the focus of modularization is decomposing a product into modules. Defining modules while establishing a platform means structuring the product's architecture according to a certain hierarchy (Marti, 2007). Essentially it divides the product architecture into a standardized part (the platform) and customized modules. (Blackstone, 2013) explained that in the platform architecture there is a grouping of products to share common parts, components and characteristics (common platform). So this kind of design can be used to reduce cost and time to market. Another definition is presented by (Meyer & Lehnerd, 1997) as "a set of subsystems and interfaces that form a common structure from which a stream of derivative products can be efficiently developed and produced." (Marti, 2007) explained that in highly modularized products, it can be advantageous to establish platforms on the level of individual modules like the example shown in figure 17.

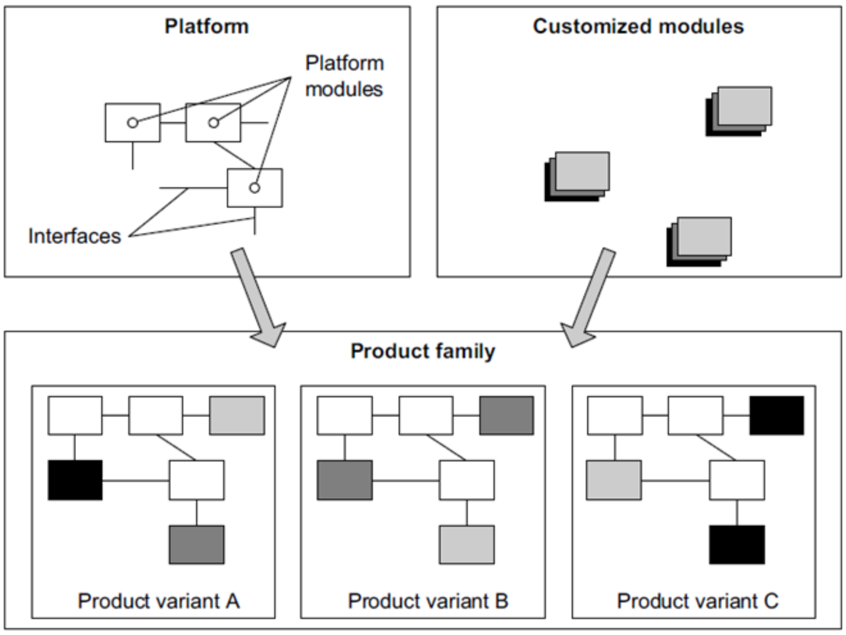


*Figure 17- Product family derived from a product platform (Marti, 2007)*

Finally, (Robertson & Ulrich, 1998) defined a product platform in a concurrent way as "the collection of assets that are shared by a set of products", not confining it to the common physical structure shared across products. So these assets fall into one of the following four categories: components, processes, knowledge, and people / relationships (Robertson & Ulrich, 1998).

Given these elements and the product generic model we propose, we can deduce that platform architecture must show:

- A large number of constraints inside each product modules (Intra-PCEP and Inter PCEP) with a high constraint density meaning that the quantity of allowed combinations is small or the dependencies between descriptive variables and families of components are strong.
- A set of constraints between modules (Inter-Module constraints) that show that only one module is linked individually with all the others with a low constraint density meaning that a high number of possible platform/modules combinations is possible.

In the example of figure 18, we have reconsidered the modular example of section 3.1.4.2 figure 15 and only modified the inter-module constraints organization. The example shows that module 2 can be considered as the product platform while modules 1 and 3 are "customized" product modules that support diversity.

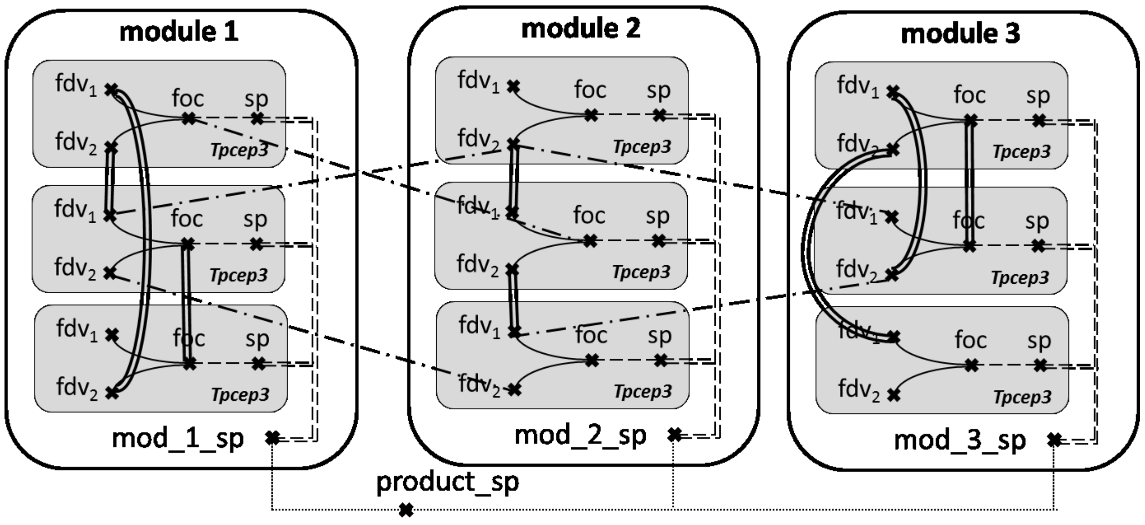


*Figure 18- Example of product model with a platform architecture*

## 3.1.5 Synthesis about product generic model for O-CPPC

Our generic model proposition for the product part in order to analyze the Concurrent Product Process Configuration gather the following key elements:

- A product is a set of modules with an evaluation variable for each criterion.
- A module is a set of Product Configuration Evaluation Patterns (PCEP) with an evaluation variable for each criterion.
- A PCEP is a set of configuration variables (or decision variables) that gathers descriptive variables and families of components and evaluation variables.
- Product configuration and evaluation constraints are present: inside each PCEP, inside each module, inside the product.
- Different kinds of patterns have been identified and describe: Product Configuration Evaluation Patterns (5 patterns), Product Configuration Constraints (4 patterns), Product Evaluation Constraints (3 patterns).
- Constraint density has been proposed in order to characterize the amount of possible solutions allowed by any constraint.

Given these elements, various examples have shown the diversity of the product aspects than can cover the CPPC problem. All this confirms that a random generation of variables and constraints in order to define a product of a CPPC problem is really far from the reality and highlight the potential interest of our proposition for comparing optimization algorithms.

## 3.2 Process configuration generic model for benchmark

The process part of the generic model is much less complex than the one relevant to the product and many already defined elements will be reused. Therefore, we follow the same kind of organization for this section and present and discuss: process modeling, process operation patterns, constraint patterns and process architectures.

### 3.2.1 Process as a set of production operations

As explained in chapter 2, authors dealing with process modeling for configuration are much less numerous. However it is possible to recall (Schierholt, 2001), (Aldanondo & Vareilles, 2008) or (Gottschalk & La Rosa, 2010) that all consider the process as a set of activities and extend the product configuration ideas to the process domain. We therefore follow the recommendations of these authors and consider that the process is a set of operations. In this work we also assume for simplicity, that the ordering and the number of operations are static. It means that there is neither OR node on the sequence nor operation activation according to a specific configured product.

Any operation gathers different resources with a given quantity during a certain amount of time in order to achieve some product production activities (sourcing, manufacturing, assembling, delivering…). Similarly, with the product domain we consider that the resources that can achieve the same kind of process are gathered in families of resources. Given a production operation to achieve, the process configuration corresponds for each operation: (i) to select a resource in a resource family (noted $for_m$) and (ii) to quantify the quantity of resources that must be used (noted $qtr_m$). We therefore firstly consider only two kinds of process configuration variables or decision variables: the family of resource and the quantity of resources that are gathered in couples for each operation. The generic process basic configuration model gathering operations is therefore { { ($for_m$, $qtr_m$) } }.

Given that our goal is to optimize the CPPC, these two operation configuration variables are strongly connected with process evaluation criteria. In our study two criteria are considered: the processing duration or cycle time (noted dur) and the processing cost that will be later aggregated with product selling price (noted sp). Other criteria like carbon footprint or quality could be considered. These criteria are computed thanks to evaluation constraints. In our bicriteria case, we consider: one for duration (noted $cst_{dur}$) and one for cost (noted $cst_{sp}$). The generic process configuration model becomes:

{ { ($for_m$, $qtr_m$, dur, $cst_{dur}$, sp, $cst_{sp}$ ) } }.

### 3.2.2 Process operation pattern

In this section, we define all variables needed to describe a generic operation. In order to associate previous operation variables with evaluation constraint, we propose to consider that each operation generates a specific work load (noted wl) for each used resource. This work load is defined for a given resource with a quantity of resource multiplied by duration (for example, a packing operation work load equal two man-month whatever the product is).

This work load permits to quantify each operation duration and each operation cost. In order to detail the operation pattern, let us first consider that the operation uses just one family of resource and the resulting single resource operation pattern shown in figure 19.

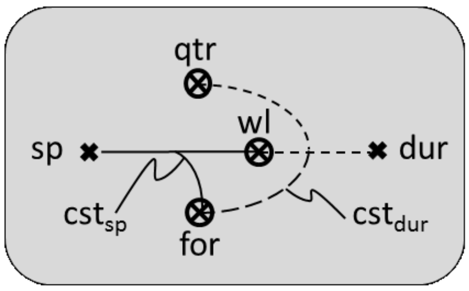


*Figure 19- Single resource operation pattern*

This pattern shows previous operation variables and the two evaluation constraints. Given a production operation to achieve, a resource can be identified in the family with a workload to achieve. Given this selected resource (for) and identified workload (wl):

- The first constraint dealing with the cost and later selling price ($cst_{sp}$) allows quantifying the operation cost (sp),
- The second constraint dealing with duration ($cst_{dur}$) allows quantifying the operation duration (dur) with respect to the resource quantity (qtr).

We will see later that the three decision variables that can be linked with product characteristics are the family of resource (for), the resource quantity (qtr) and the workload (wl).

When more than one family of resource must be considered, for example a machine and an operator as shown in figure 20, each family of resource will be associated with a single resource operation pattern and the criteria will be aggregated. Most of the time, for cost operation (noted

$sp_{op}$) criterion the aggregation is a sum (Σ on the figure 20) while for operation duration (noted $dur_{op}$) it is a max operator (max on the figure 20), other operators could be imagined. Consequently the generic operation process model becomes:

{ { ($for_m$, $qtr_m$, $wl_m$, $dur_m$, $cst_{dur}$, $sp_m$, $cst_{sp}$ ) }, $dur_{op}$ $sp_{op}$ }.

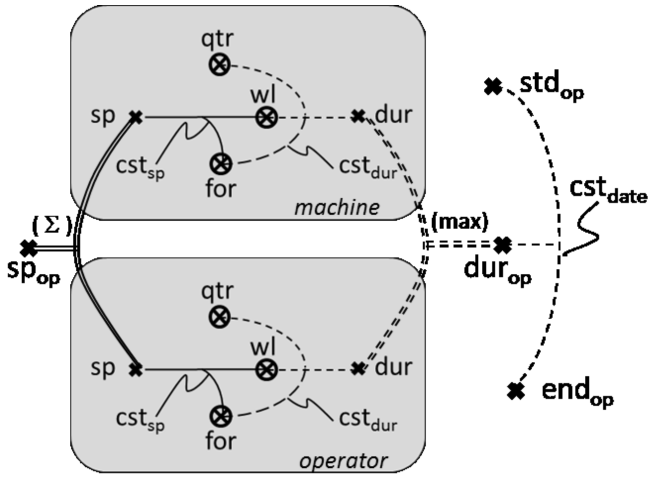


*Figure 20- Multi resource operation pattern*

Last point to discuss is relevant to the operation sequencing or ordering. This will be achieved classically by anteriority constraint. We therefore need to add to the operation pattern model, starting and ending dates (noted $std_{op}$ and $end_{op}$) variables in order to attach anteriority constraints between the operations. These two variables dates are linked with a date constraint (noted $cst_{date}$ ) stating that ending date equals starting date plus operation duration. Here we consider infinite capacity planning in order to quantify a minimum operation duration that will provide minimum production cycle time without taking into account any scheduling issues. This can be seen in the right of figure 20 and the resulting operation model is as follows:

{ { ($for_m$, $qtr_m$, $wl_m$, $dur_m$, $cst_{dur}$, $sp_m$, $cst_{sp}$ ) }, $dur_{op}$, $sp_{op}$, $cst_{date}$, $std_{op}$, $end_{op}$ }.

With the proposed elements it is possible now to finish the description of the global process generic model as a set of operation pattern models as shown in the example of figure 21. It is just necessary to add:

- Process global criteria evaluation variables as in our case: starting and ending process dates (noted $std_{pr}$ and $end_{pr}$ ) and process cost (noted $sp_{pr}$ ),
- A set of sequencing constraints between operations, this is achieved by stating temporal constraints between necessary operation starting and ending dates: most of the time: $std_{opi+1} > std_{opi}$, (noted $cst_{seq}$ )

- Aggregation constraints for criteria, in our case for a cost criterion it is most of the time a sum operation (noted $cst_{sppr}$).

Consequently, the global process model becomes:

{ { { ($for_m$, $qtr_m$, $wl_m$, $dur_m$, $cst_{dur}$, $sp_m$, $cst_{sp}$ ) }, $dur_{op}$, $sp_{op}$, $cst_{date}$, $std_{op}$, $end_{op}$ }

{ $cst_{seq}$ }, $std_{pr}$, $end_{pr}$ , $cst_{sppr}$, $sp_{pr}$ ) }.

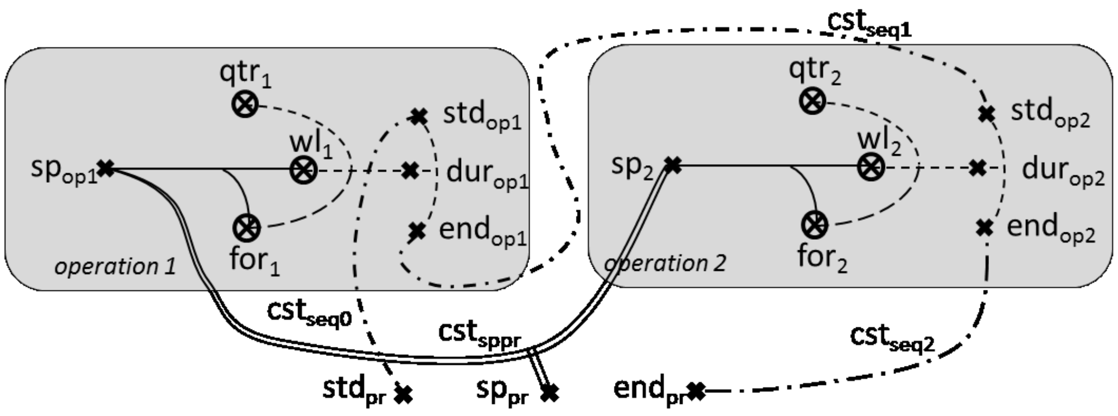


*Figure 21- Proposed CPPC process generic model for benchmark*

### 3.2.3 Process constraints pattern

Given all previous propositions, any process constraint is consequently associated with a criterion evaluation that in our case are cost and duration. Therefore, the constraints seem to be only present for criterion evaluation and could lead to the conclusion that there is no effective configuration problem. This is not the case, because in most situations you have constraints between different resources families either for a same operation (incompatibility of people for example) or between different operations (if operator A works on an operation 1 he should keep working on operation 2 for example). In the following, we just briefly comment: cost, duration and resource constraints.

#### 3.2.3.1 Process cost constraint pattern

For the cost constraint inside each operation pattern ($cst_{sp}$), the evaluation constraint patterns proposed in section 3.1.3.2 can be re-used. Mainly the first one (Tep1) which is linear with respect to the configuration variable fits very well our cost criterion with a cost more or less proportional to workload given a specific selected resource.

For the cost constraint ($cst_{sppr}$) that aggregates the cost of different operation at the process level ($cst_{sppr}$), similarly to product model there is no specific pattern as it is most often a sum calculation.

#### 3.2.3.2 Process duration constraint pattern

Inside each operation pattern, we have two constraints ($cst_{dur}$ and $cst_{date}$). The first one quantifies the operation duration with respect to: the resource selected, the workload to realize and the quantity of resources affected. Duration is more or less proportional to workload while more or less inversely proportional to resource quantity as shown in the pattern of figure 22. The second one ($cst_{date}$) is always the simple formula $end_{op} = std_{op} + dur_{op}$.

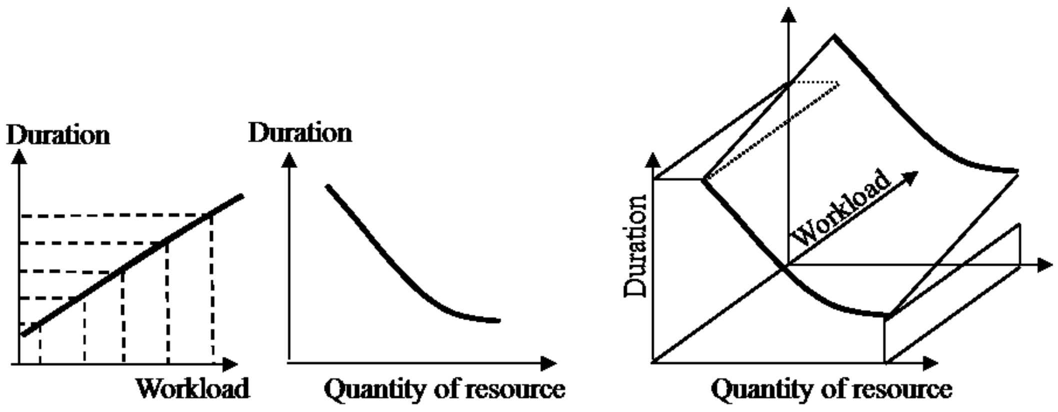


*Figure 22- Operation duration constraint pattern*

Between operations, there is no specific pattern. The already mentioned sequencing constraint ($cst_{seq}$) $std_{opi+1} > std_{opi}$ is enough as far as we don't consider any overlapping constraint.

#### 3.2.3.3 Operation resource constraint pattern

For this kind of constraint, the two configuration constraint patterns proposed in section 3.1.3.1 "Similarity or close level pattern" or "Dissimilarity pattern" can be used in order to describe a similarity or dissimilarity of resource inside or between operations. Due to the diversity of technological resources that can be used, many other kinds of compatibility constraints could be imagined.

### 3.2.4 <u>About Process architecture or structure</u>

For the Optimization of Concurrent Product and Process Configuration Problem (O-CPPC) the type of production process that we are interested in is the Assembly Line. (Grzechca , 2011) defined an assembly line as "a manufacturing process in which parts are added to a product in a sequential manner using optimally planned logistics to create a finished product in the fastest possible way". More specifically, (Grzechca , 2011) details that the "assembly" is the process of fitting together various parts in order to create a finished product, so the parts can be divided into sub-assemblies and components. Finally, (Delchambre & Rekiek, 2006) explained that the

Assembly line is a production system composed of a number of stations where the pieces are consecutively launched down the system and are moved from one station to another in an order in which they must follow according to technological restrictions. Most of the time, two main process architectures or structures are considered (Chen, et al., 2006) with respect to the operation graph structure: serial and convergence.

In the serial case, the operations or production steps are executed mainly in a serial way. This means that one operation of the process finish before the next starts and only one step is active at any one instant. This is structure is the most present when dealing with mass production as in automotive and electrical industry for example.

The convergence case is a variation of the serial configuration structure where we can find two or more operations simultaneously running and then converging to another main operation. This solution is often met when production is launched in small batches as in aeronautic or railway industry for example.

### 3.2.5 Synthesis about process generic model for O-CPPC

Our generic model proposition for the process part in order to analyze the Concurrent Product Process Configuration gathers the following key elements:

- A process is a set of operations with an evaluation variable for each criterion and two process starting and ending dates.
- An operation is a set of variables that gathers:
  - for each involved resource: (i) three configuration variables (or decision variables): resource family, quantity of resource, workload; and (ii) two evaluation variables: duration and cost in our case,
  - for the whole operation: (i) two evaluation variables: duration, a cost in our case and (ii) two operation dates: starting and ending dates
- Evaluation constraints are present: (i) for each involved resources inside each operation, (ii) then for the whole operation in order to aggregate each criterion on the different resources and, (iii) for the whole process in order to aggregate each criterion on different operations.
- Process configuration constraints are restricted to resource compatibility inside an operation or between operations.

We can see that the diversity and complexity of process configuration model is much lower than the one relevant to product. Next section will just assemble them in order to finalize our proposition for a concurrent product process configuration generic model for benchmark.

## 3.3 Coupling product process models and key characteristics

We will first show how the two models can be associated, then in a brief synthesis underlines key characteristics of the proposed model.

### 3.3.1 Coupling product and process models

Following the works published about concurrent product process configuration (Baxter, 2007), (Aldanondo, et al., 2010), (Hong, et al., 2010), (Pitiot, et al., 2013) or (Dhungana, et al., 2017), the idea is to add coupling compatibility constraints between the two models.

On the product side, the coupling constraints (noted $cst_{cpl}$) can include only families of components and/or functional descriptive variables. For example, let us consider (i) a family of components “machine tool frame” with different descriptive variables as: “size”, “material” and “weight” and (ii) an operation as “welding” and a family of resources as “operator”:

- According to the frame material, different welding competencies can be required and therefore a specific operator in the family.
- According to the frame size it is clear that the welding workload can vary.
- According to the frame weight it might be necessary to use more than one operator just to manipulate the frame.

This simple example illustrates how descriptive attributes of a family of components impacts one process operation and therefore how the two models can be connected.

On the process side, coupling constraints can only include families of resources, quantity of resources and/or workload variables. It is important to note that evaluation variables (cost and duration in our case) cannot be included in a coupling constraint.

In terms of coupling constraint patterns, the four patterns presented for the product configuration constraints (section 3.1.3.1) can be fully used as far as the definition domains of both product and process variables are ordered.

In terms of evaluation, when a criterion is present in both product and process model, as it is the case for cost or selling price, the two evaluation variables should be aggregated with a constraint (noted $cst_{sppp}$) in order to get a global product/process criterion value.

### 3.3.2 Synthesis about full product/process generic model for O-CPPC

With previous coupling constraints we can now propose our full product/process generic model for O-CPPC benchmark:

- For product:
  - A product is a set of modules with an evaluation variable for each criterion.
  - A module is the instantiation of a set of Product Configuration Evaluation Patterns (PCEP) with an evaluation variable for each criterion, cost in our case.
  - A PCEP is a set of configuration variables (or decision variables) that gathers descriptive variables and families of components and evaluation variables.
  - Product configuration and evaluation constraints are present: inside each PCEP, inside each module, inside the product.
- For process:
  - A process is a set of operations with an evaluation variable for each criterion, duration and selling price in our case, and two process starting and ending dates.
  - An operation is a set of variables that gathers:
    - for each involved resource: (i) three configuration variables (or decision variables): resource family, quantity of resource, workload; and (ii) two evaluation variables: duration and selling price in our case,
    - for the whole operation: (i) two evaluation variables: duration and selling price in our case and (ii) two operation dates: starting and ending dates
  - Evaluation constraints are present: (i) for each involved resources inside the operation, (ii) then for the whole operation in order to aggregate the different resources for each criterion and, (iii) for the whole process in order to aggregate the different operations for each criterion.
  - Process configuration constraints are restricted to resource compatibility inside an operation or between operations.
- For the whole product/process model:
  - Coupling constraints (noted $cst_{cpl}$) linking product descriptive variables and/or families of components with process resource families, quantity of resource and/or workload.
  - Product/process evaluation constraints in order to aggregate each criterion existing in the two models.

An example of such product/process model is proposed in figure 23. It assembles the two models of figures 11 and 20 with two coupling constraints and a product/process evaluation constraint:

- $Cst_{cpl-1}$ between two descriptive variables (fdv2-1 and fdv2-2) of module 2 and the three operation variables ($for_1$,$wl_1$, $qtr_1$) of operation 1
- $Cst_{cpl-2}$ between one component family ($foc_{1-3}$) of module 1, one component family ($foc_{2-31}$) of module 2 and the two operation variables ($for_2$,$wl_{21}$) of operation 2,
- $Cst_{sppp}$ that aggregates selling price criterion of product and process in a product/process selling price criterion ($Sp_{product/process}$).

Dealing with architecture, we have introduced for product: integrated, modular and platform architectures while we have mentioned serial or convergence process structures. The association of product architecture and process structure is most often serial process with platform (automotive mass production industry for example) product and convergent structure with either integrated or modular product architectures (aerospace small batches industry for example).

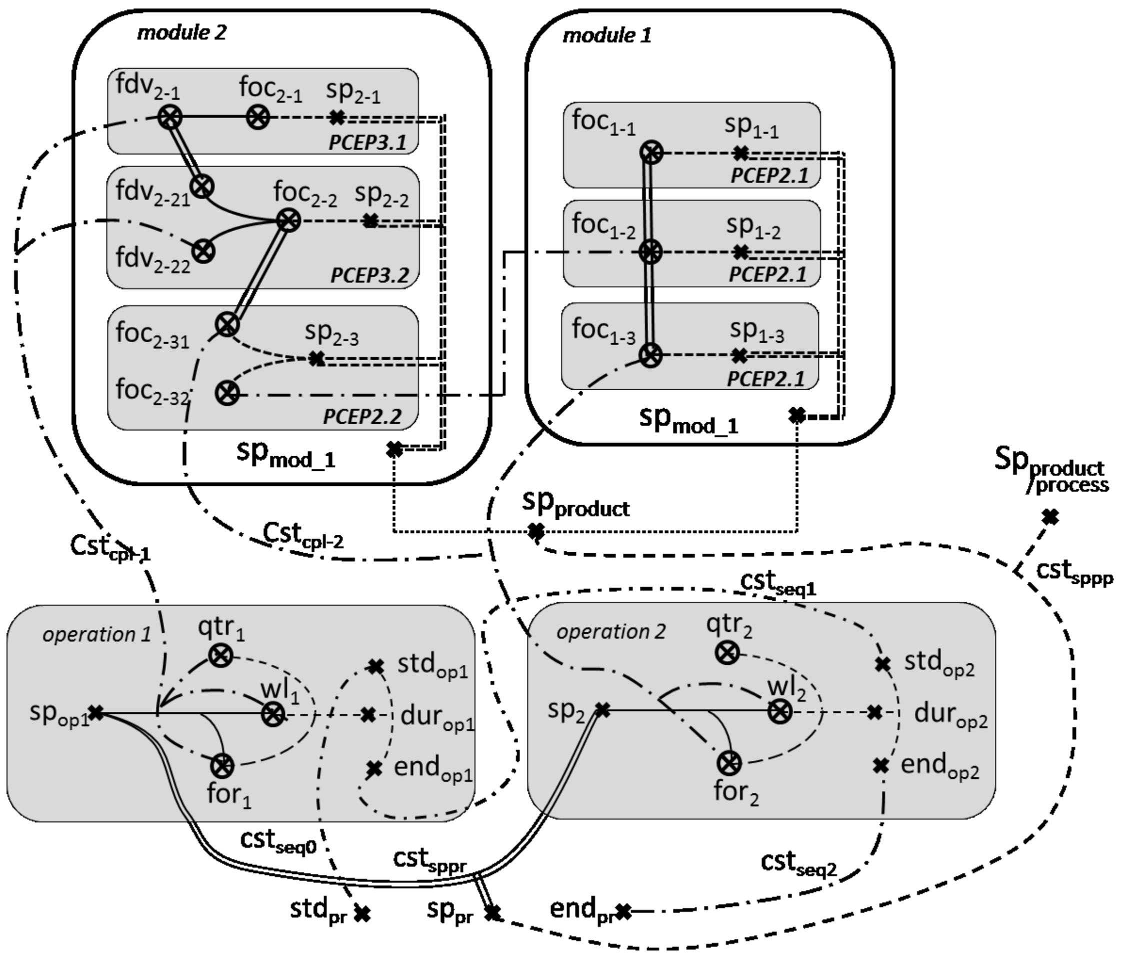


*Figure 23- Proposed CPPC product/process generic model for benchmark*

## 3.4 Proposition synthesis and key characteristics

The goal of this third chapter was to answer the question "Is it possible to propose a generic model of the Concurrent Product Process Configuration problem that can avoid case dependency when evaluating and comparing optimization methods?". Given all previous propositions, we can conclude that we now have a good base for a generic model of Concurrent Product Process Configuration problems.

Our proposition is based on two sub-model product and process with various generic patterns that capture many aspects relevant to the diversity of configuration problems. These patterns concern products, processes and constraints. It is clear that others could be added, but we deeply think that they constitute a strong base for product process configuration modeling. In order to have full confidence in these propositions, a last step of validation is necessary with some kind of confrontation with real problem modeling. However, it sounds clear to us that using our propositions in order to conduct CPPC optimization benchmarks will be always better than using random generation of variables and constraints.

The key characteristics of our generic model are: (i) for the product side: number of modules, type and number of PCEPs, number of configuration variables, number of values, (ii) for the process side: number of operations, number of resources, number of values, (iii) for product and process sides: constraint patterns and constraint density, and (iv) Product/Process architectures. These characteristics will be used in the next chapter to evaluate optimization heuristics.

# 4. Benchmark description and evaluation of CFB-EA

In this chapter, the benchmark proposed is presented, discussed and used to evaluate the CFB-EA optimization approach. The goal of the benchmark is to be able to evaluate optimization approaches on a scope of O-CPPC instances which are not case-dependent. This will allow answering our second research question "How sensitive is CFB-EA optimization method, with respect to each key characteristic of the generic model of the CPPC problem?"

The definition of the instances that constitute the benchmark follows two contradictive goals. We need to represent the diversity of existing O-CPPC problems but we also have to limit the testing effort and thus the number of instances required to evaluate correctly an optimization approach. We therefore aggregate the key characteristics proposed at the end of the previous chapter in three parameters: (i) problem size, (ii) problem constraint density and (iii) product architecture. This induces eight different instances of CPPC models that will be optimized.

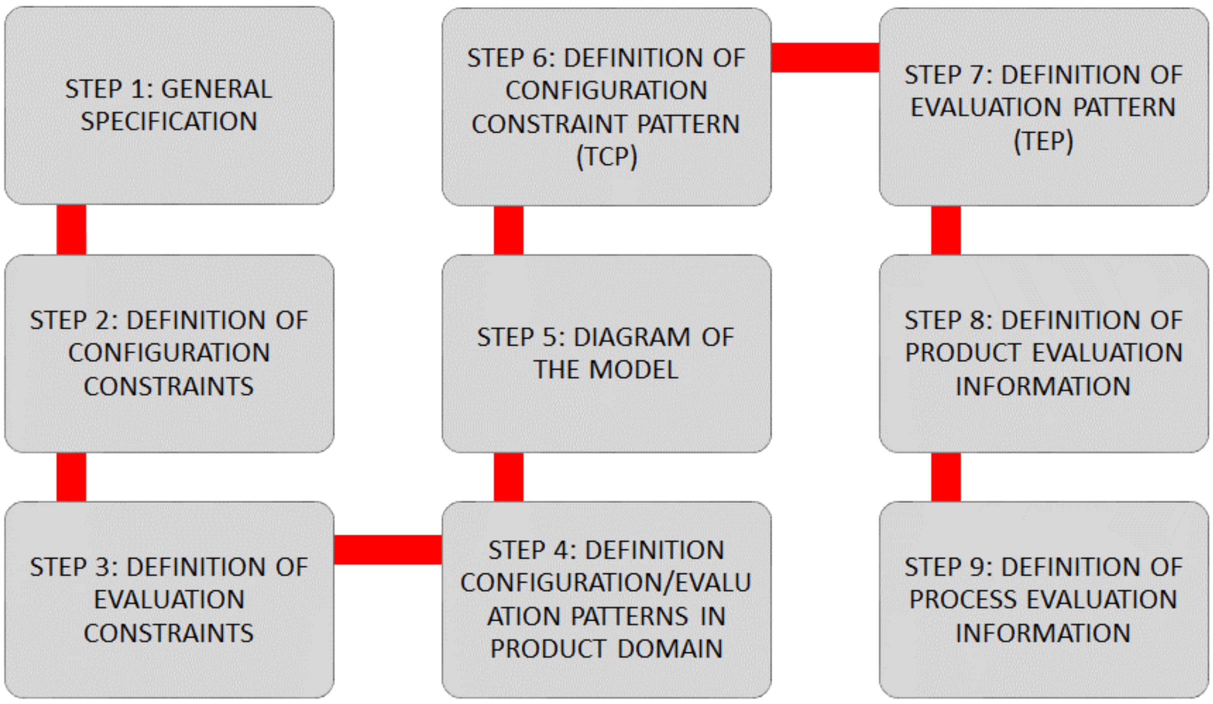
In terms of optimization, we will use an evolution of the CFB-EA algorithm presented in (Pitiot, et al., 2014) and evaluates how it behaves with respect to the eight previous instances and three previous problem characteristics. This evolution gathers some detailed computation improvements which will not be explained in this thesis: better and faster evaluation of individuals and faster crossover operator. This explains the lower computation time when compared with results published in (Pitiot, et al., 2014) and (Pitiot, et al., 2019).

This chapter is organized as follows. Firstly, we detail the process used to generate model instances and the eight instances (cases). Secondly, we present the optimization evaluation process. Thirdly, we analyze those results and conclude about key characteristics impacts on optimization.

## 4.1 Definition and generation of problem cases

### 4.1.1 Model generation procedure

In order to get problem cases, we develop a software called "CPPC model generator" that instantiates the generic model presented in the previous chapter. This generator allows fulfilling all various parameters specified in generic model (i.e. the number of modules, the type of patterns and their parameters, etc.). To support this fastidious task, the model generator implements a procedure to create an instance of an O-CPPC model. This procedure is shown in figure 24 and works as follows:



STEP 1: GENERAL SPECIFICATION
STEP 2: DEFINITION OF CONFIGURATION CONSTRAINTS
STEP 3: DEFINITION OF EVALUATION CONSTRAINTS
STEP 4: DEFINITION CONFIGURATION/EVALU ATION PATTERNS IN PRODUCT DOMAIN
STEP 5: DIAGRAM OF THE MODEL
STEP 6: DEFINITION OF CONFIGURATION CONSTRAINT PATTERN (TCP)
STEP 7: DEFINITION OF EVALUATION PATTERN (TEP)
STEP 8: DEFINITION OF PRODUCT EVALUATION INFORMATION
STEP 9: DEFINITION OF PROCESS EVALUATION INFORMATION

*Figure 24- Main procedure to generate a case in CPPC model generator*

- In step 1, the generation specifications that allow to globally describe the main characteristics of the problem are inputted:
  - Product architecture: platform, modular, integrated.
  - Process architecture: serial, convergence.
  - Model size: small, medium, intermediate, large.
  - Constraint density: low, medium, high.

Given these elements, the quantity of: product modules, process operations, product and/or process evaluation criteria can be chosen.

- In step 2, consequently the quantity of configuration constraints with types and density can be input for:
  - Product: Intra-PCEP constraint, Intra-module constraint, Inter-module constraint.
  - Process: Inter-operation constraints.
  - Product and process: Coupling constraints.

- In step 3, a quantity of evaluation constraints for both product and process is proposed and validated. In our case we have only duration and selling price evaluation variables:
  - For selling price, the product architecture with the number of: modules, PCEP and operations allow to deduce possibilities for these quantities.
  - For duration, the number of operations and the process architecture allow to deduce possibilities for these quantities.
- In step 4, for each module and each PCEP, the type of pattern is selected.
- In step 5, the resulting product/process model structure with all variables and constraints can be established and validated by the user.
- In step 6, for each configuration constraint, a constraint pattern is selected for product, process and coupling.
- In step 7, for each evaluation constraint, an evaluation pattern is selected for product and process.
- In step 8, product evaluation parameters are inputted for each product evaluation pattern, in our case selling price, mainly the average value with an interval of variation of the selling price variable,
- In step 9, process evaluation parameters are inputted for each process evaluation pattern, in our case selling price and duration, mainly averages and intervals for the selling price and duration variables.

This procedure has been followed eight times in order to define eight model instances or cases that allow evaluating the main characteristics of an O-CPPC model: The product architecture, the model size and the constraints level or density. All cases derive from a reference case which is a platform architecture model.

### 4.1.2 Main characteristics of the reference platform model case O-CPPC

In this section, we describe the main characteristics of the reference case which is the platform architecture model showed in figure 25. This reference model gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 24 configuration variables in product side (14 fdv and 10 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).

- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for the whole model around $10^{23}$).
- 9 PCEP patterns (three patterns for each module),
- 26 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 26 evaluation constraints (constraints needed to compute selling price and cycle time).

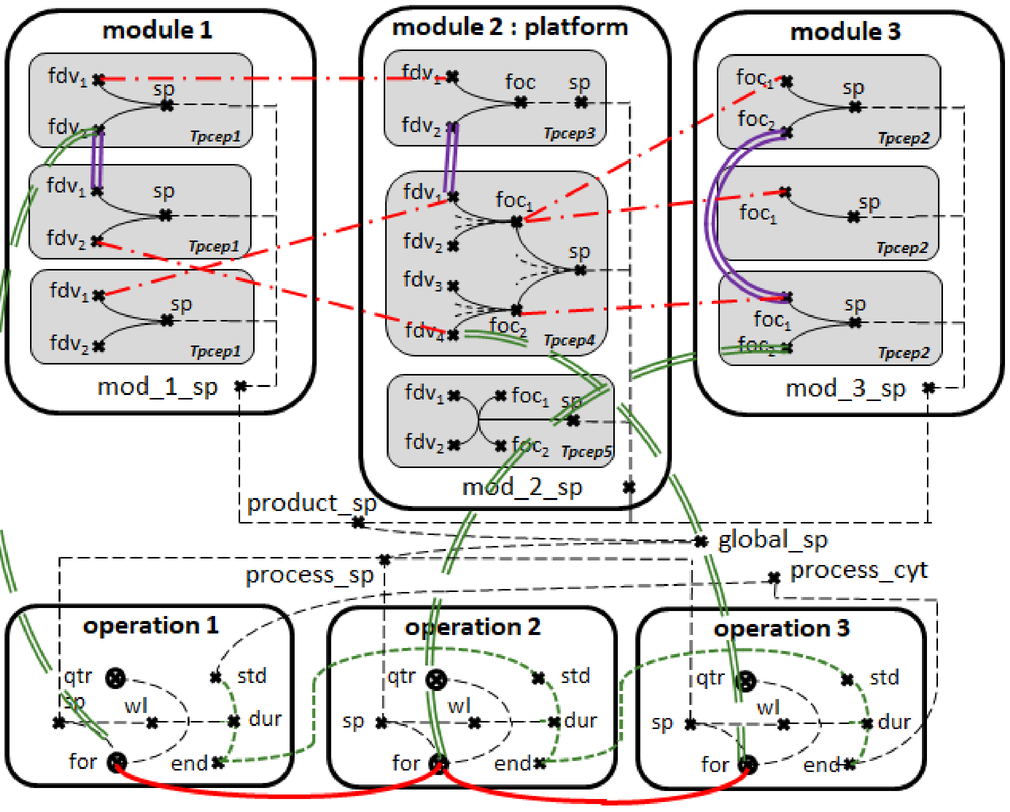


*Figure 25- Reference case, the platform model*

One module, module 1, has a functional description with only Tpcep1 pattern (only functional description and selling price variables). Another module, module 3, has component description with only Tpcep2 patterns (only family of components and selling price variables). While the platform module, module 2, gathers a selection of physical-functional description with mixed patterns. Each module is linked to one operation by a coupling configuration constraints and each operation is linked to another operation by a configuration constraint. For all reference model details please consult annex 1.

Considering this reference model as a basis, we now describe how the seven other model instances are going to be built with respect to the three characteristics: model size, constraint density and product architecture.

### 4.1.3 Model Size

The size of the model is a key point for optimization evaluation. It corresponds to the size of the search space to investigate and is primarily associated with the number of variables and the size of their domains. The reference model has 30 variables with 6 values on their domains which is a good size (around $10^{23}$ without constraints) to investigate quickly properties of evaluation approaches in a reasonable computing time (around some hours of calculations). Real-world models could be obviously bigger but we assume that a detailed representation of a large and complex product/process would still remain under one or two hundred variables (assuming between 2 to 20 possible values and not only binary variables).

In our approach, the user restrains the model size by entering his/her requirements and process filtering that decreases the solution space size. The resulting size could therefore greatly vary according to the number of inputted requirements. We consequently select four cases to study this model size characteristic with four model size values: 100, 60, 30 and 15 variables. Those cases could be various sizes corresponding to different reduction made by user according to his/her requirement on product/process. The four cases characteristics are detailed in table 2. Particular attention was paid to respect distribution and density of all constraints in order to be able to evaluate only the size impact. For these four instances, the following ratios are roughly constant:

- Number of configuration constraints / number of variables: between 0.5 and 1, that provides a number of constraints of 12, 26, 51 and 82.
- Number of modules / number of variables: between 0.1 and 0.2, that provides a number of modules of 3, 7 and 10.
- Number of operations / number of variables: between 0.1 and 0.2 that provides a number of operations of 3, 7 and 10.

Similarly, the constraints densities are similar or close (around 50% of constraints with high level (inter-modules constraints) and 50% with medium level (other constraints)) and the constraints distribution between various types of constraints (intra-module, inter-module, coupling, etc.).

| Size | Quantity of Variables | Quantity of Modules | Quantity of Operations | Quantity of Configurations Constraints |
|---|---|---|---|---|
| 1. Small | 15 | 3 | 3 | 12 |
| 2. Medium | 30 | 3 | 3 | 26 |
| 3. Intermediate | 60 | 7 | 7 | 51 |
| 4. Large | 100 | 10 | 10 | 82 |

*Table 2- Model Sizes*

### 4.1.4 Configuration constraints Density

Constraints density or hardness is delicate issue to evaluate. It could be linked to the number of constraints, the number of allowed tuples by constraint or their distribution between variables. The number of constraints and the number of tuples by constraint impact in same way the number of feasible solutions. Impact of distribution of constraints in the model or distribution of constraints with different levels would be more difficult to evaluate. As already said, we consider configuration constraint density as the ratio of the number of forbidden tuples excluded by the constraints divided by the number of possible tuples without any constraint. We defined three levels of constraint density: low, medium and high. The low level corresponds to 20% of ratio of forbidden tuples, medium to a level of 50% and high to a level of 80%. The configuration constraints density is thus evaluated on the same platform architecture model with 26 constraints and 30 variables. The three cases differ only by the constraint's density level of each constraint with three levels low, medium and high.

### 4.1.5 Product architecture

As we presented in chapter 3, we are going to use the three typical product architectures: Platform, Modular and Integrated. The idea is to evaluate if the product architecture impacts optimization performance. The difference between these architectures relies on: (i) the distribution of product configuration constraints between: Intra-PCEP, Inter-PCEP, Inter module constraints and (ii) their constraint density.

In the platform architecture, a module is the platform and the two other modules are linked on it. Following the idea that constraints in a module are stronger than between modules, the 12 constraints in modules (intra-PCEP and inter-PCEP) have a higher constraint density; while constraints between modules have a medium density.

Modular architecture derives from platform architecture one only by distribution of inter-module configuration constraints. Each module is linked to the two others by two configuration constraints.

Integrated architecture has the same constraints distribution than modular architecture. The difference is that constraints between modules are stronger. To keep the same constraints density level in the whole model, we put some intra-module constraints to medium density. In such way, for all architecture there is the same number of constraints with high (15) and medium (11) density level.

For all these product architectures, we assume a serial process architecture in order to isolate and quantify only the effect of product architecture modifications.

### 4.1.6 Synthesis about O-CPPC benchmark

The eight model instances for our benchmark are presented in Table 3. Each of them will be optimized with CFB-EA algorithm in the next sections.

| | **Size** | **Quantity of Variables** | **Architecture** | **Constraint density** | **% of forbidden tuples** |
|---|---|---|---|---|---|
| 1 | Small | 15 | Platform | Medium | 50 |
| 2 | Medium | 30 | Platform | Medium | 50 |
| 3 | Intermediate | 60 | Platform | Medium | 50 |
| 4 | Large | 100 | Platform | Medium | 50 |
| 5 | Medium | 30 | Modular | Medium | 50 |
| 6 | Medium | 30 | Integrated | Medium | 50 |
| 7 | Medium | 30 | Platform | low | 20 |
| 8 | Medium | 30 | Platform | high | 80 |

*Table 3- O-CPPC benchmark model instances*

## 4.2 Optimization experimental plan

This section presents optimization experiments on previous cases using evolutionary algorithm (CFB-EA). The EA optimization algorithm is implemented in C++ programming language and interacts with the filtering system Cofiade (Vareilles, et al., 2012) coded in Perl language. The indicated time consumption is the processor time spend by all processes (EA + filtering engine). The real time spend by optimization could be divided thanks to the parallelization of filtering engines (around 95% of time consumption correspond to the filtering process). In this section, we present the setting used for the evolutionary algorithm and the metrics used for experiments.

### 4.2.1 Metrics for experiments

For a multiobjective problem, the user expects an efficient and diversified set of solutions in a reasonable lapse of time. To evaluate the algorithm results, we used the hypervolume metric defined in (Zitzler & Thiele, 1998). It measures the hypervolume (HV) of the space dominated by a set of solutions, as shown in Figure 26, where two criteria are considered and the HV is the surface inside the grey solid line. HV allows to evaluate both convergence and diversity proprieties. The fittest and most diversified set of solutions is the one that maximizes hypervolume.

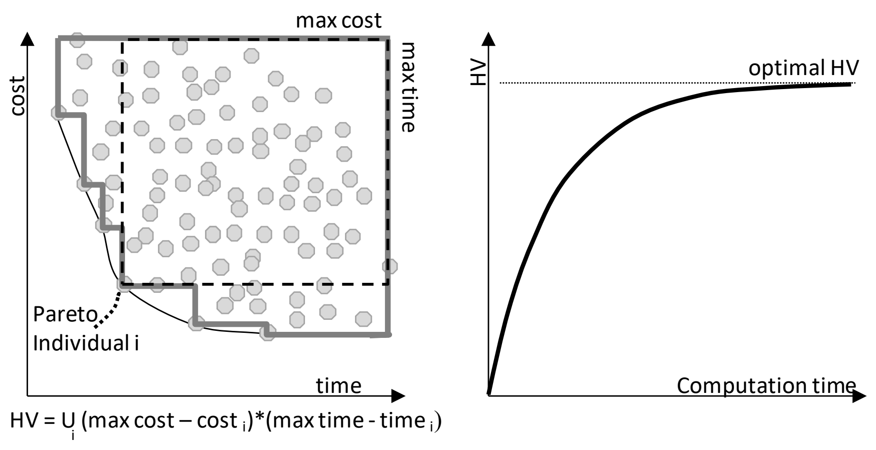


*Figure 26- Hypervolume with the two criteria time and cost*

### 4.2.2 Evolutionary settings

The CFB-EA method has six parameters and a stopping criterion. For this work, we investigate a small experimental plan on reference case (platform model) and we keep the same settings with all other models:

- Probability of crossover: 0.9
- Probability of mutation of a parent: 0.5
- Probability of mutation of each gene of a selected parent: 0.1
- Backtrack limit of filtering engine: 30. This correspond with the number of allowed backtrack for each individual before giving-up.
- Population size: 100
- Archive size: 150.

Archive and population sizes are adjusted according to the size of the model for larger cases. The stopping criterion is a strict limit of computation time in seconds. This time is adapted at each case in order to let the algorithm converge.

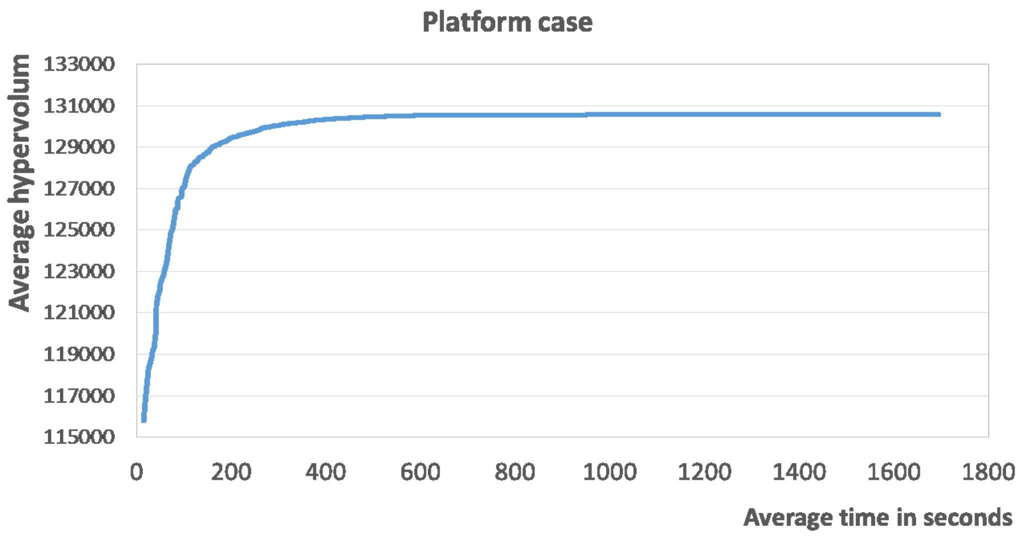
Like any metaheuristic, CBF-EA uses pseudo-random process. Consequently, the algorithm has to be launched more than one time on a same model in order to get consolidate results. In this work, every test is launched 5 times (5 runs). The average hypervolume (HV) and the relative standard deviation (RSD) over the 5 runs are computed. In each test, we compare the average time needed to reach the final average HV value noted $HV_{final}$. The $HV_{final}$ value is obtained when there is no improvement of average hypervolume.

In order to study the HV evolution, we also compare the average computation times needed to reach 99% and 99.9% of the average $HV_{final}$.

### 4.3 Evaluation of existing approach on benchmark

In this section, we present the results of the experimental plan proposed in section 4.1.6 relevant to size test, constraints density test and architectural test.

For a first illustration, figure 27 shows the evolution of average HV on platform case and associated table. The curve represents evolution of average HV for the 5 runs.



| Case | $HV_{final}$ | | | | 99,9% of $HV_{final}$ | | | | 99% of $HV_{final}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | time (sec.) | RSD time % | HV | RSD of HV % | time | RSD time % | HV | RSD of HV % | time | RSD time % | HV | RSD of HV % |
| Plaform | 1696 | 28,579 | 130587 | 0 | 502 | 0,566 | 130452 | 0,030 | 189 | 2,003 | 129286 | 0,184 |

*Figure 27- Reference case: average HV evolution with respect to time*

After a quick improvement, the average HV stagnates and slowly converges to $HV_{final}$. Around 99% of $HV_{final}$ is reached in 189 seconds and 99.9% in 502 seconds while $HV_{final}$ is reached in 1696 seconds. At the beginning, the RSD of time are limited (less than 2% of average time). This indicates that all the runs reach 99% of $HV_{final}$ in approximatively the same computation time, whereas, the RSD of time to reach $HV_{final}$ is larger (28%). Some runs (mainly one of the 5 runs) have troubles to refine Pareto front to its final value. The times needed to find $HV_{final}$ for each of the five runs are around 900, 1000, 1300, 1400 and 2200 seconds. However, all the runs found the same final for $HV_{final}$ (RSD of HV is 0%). We can guess that this value is probably the optimal one.

As all cases presented in this section will have different HV values, it would not be significant to compare their HV evolution in a same graph. We will compare only their convergence rates (time to reach $HV_{final}$, 99.9% of $HV_{final}$ and 99% of $HV_{final}$).

The stopping criterion is a strict time limit, in this case 3600 seconds (1 hour). It corresponds to around 300 generations, 16 500 individuals generated and 273 000 filtering (averagely 75 filtering per second). The main characteristic of our constrained optimization approach is to maintain feasibility of individual during their construction or modification (crossover, mutation or initialization). At each modification of an individual (i.e. an instantiation of a decision variable), domains of all remaining variables are checked by filtering engine. When a domain of remaining variables become empty, the individual is unfeasible. A limited backtrack on previous choices is then launched to restore feasibility.

As all domains are checked at each modification, a strong specificity of our approach is that there is very few backtracks during evolutionary operators. In this case, there is only 0.44% of backtrack by individual. This is also due to the crossover operator behavior: the restrained crossover selects parents close in search space; then a uniform crossover is initiated on selected parents (crossover probability of 50% for each gene). It is applied gene by gene in a random order. After every instantiation, domains of remaining genes are checked: if a domain is empty, the backtrack is launched; but also, if a domain is reduced to one state, this state is automatically selected. This behavior preserves coherent genes combinations from parents in their children.

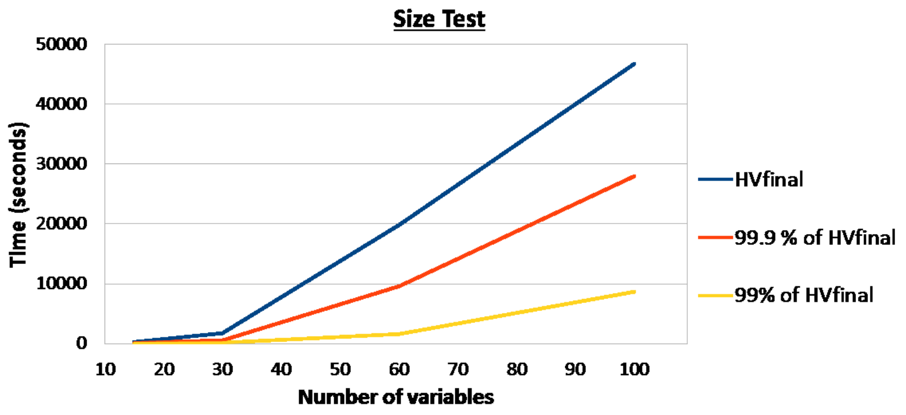
But the main drawback of this behavior is that many modifications are avoided and sometime the resulting individuals are similar to the original ones (i.e. children are similar to their parents). In this case, the rate of useless crossover (crossover that leads to the same individual) is very high with around 38%.

We will study in next sections how those rates evolve according to size, constraints density or architecture.

### 4.3.1 Model size evaluation

The results obtained with the four model size (small: 15 variables, medium: 30 variables, intermediate: 60 variables and large: 100 variables) are presented in the Figure 28. The horizontal axis corresponds to the number of variables and the vertical one with the computation time.

Logically, the time consumption is strongly affected by model size. Indeed, the number of possible solutions grows exponentially with the number of variables. Respectively, cases with 15/30/60/100 variables correspond to more than $10^{11}/10^{23}/10^{46}/10^{77}$ possible combinations without taking into account constraints and approximatively $10^{8}/10^{16}/10^{31}/10^{53}$ feasible solutions when constraints are taken into account (we will discuss in next section relations between constraints density and feasible search space).



| Size | $HV_{final}$ | | | | 99,9% of $HV_{final}$ | | | | 99% of $HV_{final}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | time (sec.) | RSD time % | HV | RSD of HV % | time | RSD time % | HV | RSD of HV % | time | RSD time % | HV | RSD of HV % |
| Small | 294 | 22,785 | 121728 | 0 | 109 | 1,560 | 121605 | 0,061 | 59 | 2,754 | 120492 | 0,584 |
| Medium | 1696 | 28,579 | 130587 | 0 | 502 | 0,566 | 130452 | 0,030 | 189 | 2,003 | 129286 | 0,184 |
| Intermediate | 19842 | 0,080 | 637302 | 0,053 | 9643 | 0,203 | 636644 | 0,128 | 1617 | 0,554 | 630876 | 0,419 |
| Large | 46782 | 0,044 | 1601750 | 0,271 | 27985 | 0,071 | 1600300 | 0,254 | 8692 | 0,158 | 1586070 | 0,213 |

*Figure 28- Model size effects on computation times*

In the small case, every run found the same $HV_{final}$ (supposed optimal value) in averagely 294 seconds (5 minutes) because relative standard deviation (RSD of HV) is 0%. Likewise, in medium case, the supposed optimal is reached in averagely 1696 seconds (28 minutes). Time gaps between runs are relatively significant with RSD on time around 28%. Nevertheless, those time consumptions and sizes are relevant for a decision aiding process for small product/process like a personal computer, a kitchen or a car (see section 1.1.4.3 on optimization issues).

With large or intermediate size, the times needed to reach $HV_{final}$ (46782 seconds/12.9 hours for large case and 19842 seconds/5.5 hours for intermediate case) are clearly much larger. Notice that some runs don't find the supposed optimal value for $HV_{final}$ before the selected time limit (time limit: 12 hours for intermediate case and 24 hours for large case). But provided $HV_{final}$ values are really close from each other given the low values of RSD (0.053% for intermediate size and 0.271% for large size).

Time consumptions in intermediate and large cases remain compatible for big products/processes (for example, an aircraft or a sealing boat). Assuming a very large product/process model of 200 variables, its sounds quite to reasonable to consider 100 customer requirements that provide a problems size to optimize around 100 variables. Our experimentations with 60 to 100 variables can be consequently considered as good representatives. Furthermore, real computation time could be significantly reduced thanks to a parallelization of filtering.

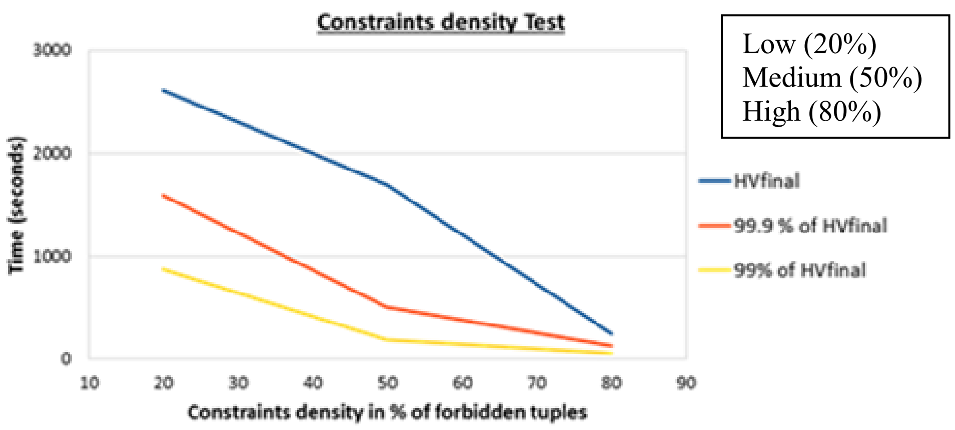
These larger cases have another property different from smaller cases: their Pareto front have more solutions than the archive size. That leads to an $HV_{final}$ value that fluctuates. Indeed when there are more Pareto-optimal solutions than the size of the archive, the algorithm make a non-deterministic selection based on distance to the k-nearest neighbor (see SPEA2 description on (Zitzler, et al., 2002)). The small values of RSD on time for intermediate and large cases indicate that all runs have reached an average $HV_{final}$ value and fluctuate around this value. Larger archive size could be selected in such cases but it will reduce convergence speed.

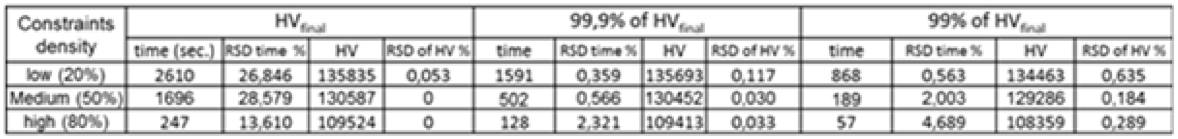
Concerning detailed indicators, size of the case to optimize significantly impacts the useless crossovers rate. It goes from 50% for the small case to 8% for the larger case. The more the chromosome of individuals is large, the more the useless crossover rate is reduced.

Another interesting observation is that, in all cases, a good approximation of the final Pareto front, for example let us consider 99% of $HV_{final}$, is found relatively quickly. It takes roughly between 10% and 20% of the time to reach 99% of $HV_{final}$. Furthermore, this is obtained with a reasonable RSD on HV (values less than 0.6% are reported). This characteristic will be used in next chapter to propose an improvement of optimization approach. The proposed improvement could also solve the archive size problem on large cases.

### 4.3.2 Model constraints density evaluation

The results obtained with three constraints density level (low: 20% of forbidden tuples, medium: 50% of forbidden tuples, high: 80% of forbidden tuples) are presented in Figure 29. The horizontal axis corresponds to the average percentage of forbidden tuples and the vertical one corresponds as before with the computation time.





| Constraints density | $HV_{final}$ | | | | 99,9% of $HV_{final}$ | | | | 99% of $HV_{final}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | time (sec.) | RSD time % | HV | RSD of HV % | time | RSD time % | HV | RSD of HV % | time | RSD time % | HV | RSD of HV % |
| low (20%) | 2610 | 26,846 | 135835 | 0,053 | 1591 | 0,359 | 135693 | 0,117 | 868 | 0,563 | 134463 | 0,635 |
| Medium (50%) | 1696 | 28,579 | 130587 | 0 | 502 | 0,566 | 130452 | 0,030 | 189 | 2,003 | 129286 | 0,184 |
| high (80%) | 247 | 13,610 | 109524 | 0 | 128 | 2,321 | 109413 | 0,033 | 57 | 4,689 | 108359 | 0,289 |

*Figure 29- Constraint density effects on computation times*

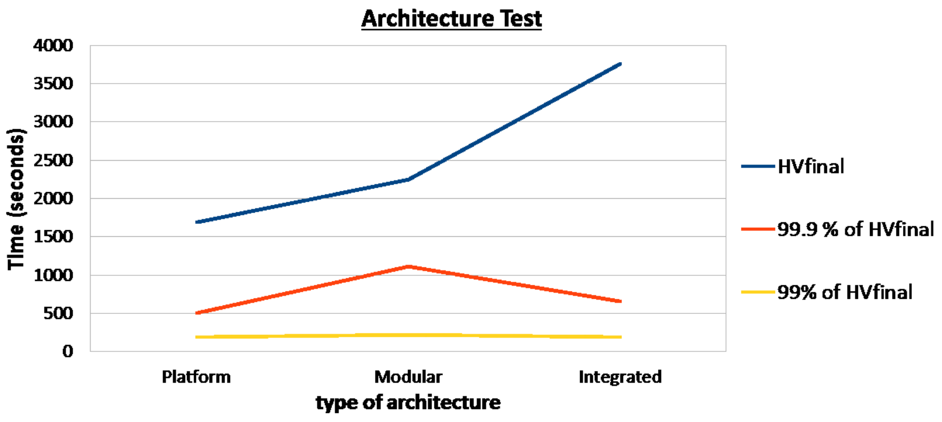
When comparing the time required to obtain $HV_{final}$, the constraints density have a major impact on time consumption. When the constraint density increases, the average percentage of forbidden tuples increases also (the problem is logically more and more constrained) and the computation clearly decreases. We confirm the known result; a more constrained problem is quicker to optimize. The most constrained problem (80% of forbidden tuples) reaches $HV_{final}$ in averagely 247 seconds with a RSD of 0% (supposed optimal), while in less constrained problem (20% of forbidden tuples), it takes averagely 2610 seconds to reach $HV_{final}$ with a RSD of 0.053%.

The constraints density level is directly linked with the number of feasible solutions and thus to the size of the search space. For a given model size (number of variables and size of their domains), the more a problem is constrained, the less it has feasible solutions. The exact number of feasible solutions can't be computed easily. It depends on the quantity of tuples of constraints but also to the distribution of constraints between variables. This constraint density test uses the platform medium size model. It has 30 variables with 6 values and 26 configuration constraints. We can have an approximation of feasibility scale of the three level of constraints density with a simpler fictive model: if 27 variables where linked one by one sequentially by constraints and the 3 remaining variables where unlinked to others, the ratio of feasibility (number of feasible solution over all possible combinations without taking into account constraints) is respectively around $10^{-3}$, $10^{-8}$ and $10^{-19}$ for high, medium and low constraints density. We checked this ratio on high constraint density case by a random sampling of solution and we found the same magnitude order: 0.002 % of feasible solutions (882 feasible solutions for 426549 random combinations tested). The constraints density level can also be evaluated thanks to the backtrack rate observed in each case. It is respectively 0.04% / 0.44% / 14.12% for high / medium / low constrained cases. Even if constructing an individual is longer for more constrained cases, the time needed to investigate feasible search space is clearly reduced.

The time consumptions needed to reach 99% of $HV_{final}$ are of the same order of magnitude than the times reported for the previous size evaluation (between 10 and 30% of time needed to found $HV_{final}$). As in previous experimentations, the dispersion of all results is very low with RSD for both time to reach 99.9% or 99% of $HV_{final}$ (less than 3%) or value of $HV_{final}$ (less than 0.4%).

### 4.3.3 Product architecture evaluation

The results obtained with the three product architectures (platform, modular, integrated) are presented in the Figure 30. The horizontal axis corresponds to the architectures and the vertical one with the computation time.



| Architecture | $HV_{final}$ | | | | 99,9% of $HV_{final}$ | | | | 99% of $HV_{final}$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | time (sec.) | RSD time % | HV | RSD of HV % | time (sec.) | RSD time % | HV | RSD of HV % | time (sec.) | RSD time % | HV | RSD of HV % |
| Platform | 1696 | 28,579 | 130587 | 0 | 502 | 0,566 | 130452 | 0,030 | 189 | 2,003 | 129286 | 0,184 |
| Modular | 2250 | 46,064 | 129364 | 0 | 1112 | 2,524 | 129233 | 0,088 | 223 | 1,099 | 128062 | 0,475 |
| Integrated | 3763 | 40,075 | 128363 | 0,034 | 657 | 4,567 | 128229 | 0,083 | 191 | 2,295 | 127054 | 0,416 |

*Figure 30- Architecture effects on computation times*

When considering the time required to obtain $HV_{final}$ value, the platform architecture is the fastest (1696 seconds) followed by modular (2250 seconds) and integrated (3763 seconds) architectures.

The gap between times needed to reach $HV_{final}$ in modular and integrated cases aren't fundamentally significant but we have to keep in mind that differences between those three cases consist mainly in different constraint organization and different distribution of constraints density. For example, if we consider the distribution of low constraints density, the same number of strong constraints (i) between modules for integrated architecture or (ii) intra-modules for modular architecture haven't a great impact on computation time when compared with the problem size or the constraints density.

These results on platform model suggest that an architecture founded on a central module is easier to investigate for an evaluation algorithm. Variables of the platform module are more connected by constraints that those of other modules. The combinations for this architecture tend to give some kind of structure to the search space more properly than in the two other cases. The evolutionary algorithms that handle combinations of genes take benefits of this structuration and reach more quickly interesting areas.

In a modular architecture, constraints are more "scattered" in the model. This leads to the same effect as a lower constraint level and thus a higher time consumption to reach final HV.

Integrated case is clearly the most difficult case to obtain final HV. The backtrack rate and useless crossover rate are significantly higher with 1.87% and 47% (against 0.24% and 40% for modular case). Some interesting individuals are found quickly (i.e. time to reach 99% and 99.9% are lower than in modular case) but it takes a long time to refine final Pareto front.

## 4.4 Result synthesis

The goal of this chapter was to evaluate an optimization approach of Concurrent Product Process Configuration with problem instances that were not case-dependent. The generic model of chapter 3 has allowed us to generate a benchmark of eight CPPC problems organized in an experimentation plan with respect to three key characteristics, (i) problem size, (ii) constraint density and (iii) product architecture. Then, these eight problems have been optimized with the CFB-EA algorithm and compared.

In terms of case-dependency, we can conclude that the proposed generic model that takes into account many aspects of the diversity of configurable product is a good way to avoid case-dependency when comparing optimization techniques. Two ideas for future works on this model are on one side to publish on the web our problem generator, and on the other side, to model new industrial cases.

In terms of optimization results comparison with respect to the three key product/process characteristics, there is no "breaking results". We have noticed the conventional results relevant to problem size and constraint density already published in [Pitiot et al., 2014]. CFB-EA shows a significant ability to combine individuals with few backtracks. Globally, it shows time consumption relevant for the addressed decision aiding process.

We would have thought that the product architecture impact would be clearer. Given our result, we can just conclude that platform architecture fits better optimization than modular and integrated architectures. This is consistent with the fact that a CSP problem well-structured (around a platform module) is easier to optimize while with a collection of small interconnected CSP (modular and integrated cases), optimization process has trouble to refine Pareto front. Comparison between modular and integrated cases confirms that when constraints are more distributed between small interconnected CSP (integrated case), it is harder to optimize finely.

Last important point for the next chapter; all evaluated cases have shown that 99% of $HV_{final}$ was reached quite quickly by all experimentations. An average of 16% of the time to reach $HV_{final}$ is necessary to get 99% of this final or very close to supposed optimal hypervolume value. This remark is the basis of the EA improvement proposed in the next chapter.

# 5. Improvement of existing approaches on benchmark

The present chapter is based in the next article:

*Paul Pitiot, Luis Garces Monge, Michel Aldanondo, Elise Vareilles & Paul Gaborit (2019) Optimisation of the concurrent product and process configuration: an approach to reduce computation time with an experimental evaluation, International Journal of Production Research, DOI: 10.1080/00207543.2019.1598598*

In order to answer to our third research question: "Is it possible to reduce the computation times of CFB-EA and other conventional EA approaches?" we propose in this chapter an improvement of CFB-EA called CFB-EA+ recently published in (Pitiot, et al., 2019). The main idea is first to quickly compute a rough Pareto of solutions, then ask the user to select an area of interest, and finally to launch a second computation on this restricted area.

In following sections, we make a brief review of associated concepts, a presentation of proposed approach, its interests and tuning of its parameters. Then, the proposed CFB-EA+ is presented with experimental results.

## 5.1 Possible improvement to the EA and computation time reductions

The CFB-EA algorithm shows a conventional behavior similar to most population-based optimization approaches and most EAs: after an initial fast improvement (due to the fitness function that maximizes solution dispersion on the Pareto front), performance stagnates before slowly coming closer to optimal values. A challenging aim is thus to find a way to avoid the previous stagnation sub-step. Given this purpose, two kinds of approaches, both of which are multi-stage, can be investigated. The first stage is almost always an EA, while the second stage can be either a similar EA or another technique, which is most often a stochastic local search (SLS).

SLS algorithms (Hoos & Stützle, 2004) move from solution to solution in the search space by applying local changes until they find a supposed optimal solution or reach a time bound. In multiobjective contexts, the idea of mixing population-based approaches with an SLS algorithm is to benefit from both the quick and global improvements of the first EA and then use the SLS algorithm to improve search around founded solutions. It leads to a well-converged and diversified final result (Blot, et al., 2018).

The idea around EA-based multi-stage optimization is to avoid tackling the whole problem size and complexity in a single shot. Two kinds of ideas can be found in the literature. The first ones try to reduce the number of criteria in a single optimization shot with some kind of criteria distribution or association to different optimization stages. The second ones try to reduce the size of solution spaces in a single optimization shot with some kind of solution space decomposition and allocation to optimization stages. For example, (Ascione, et al., 2016) distribute optimization criteria in a three-stage optimization approach in a study of energy retrofitting of hospital buildings. A similar approach can be seen in (Hamdy, et al., 2013), also for building optimization. Dealing more with solution space decomposition (Ji, et al., 2017) suggest using feasibility rules to quickly find a valid part of the solution space, then, once a solution is found, the investigated solution space is expended thanks to coevolution. Of course, approaches mixing criteria and solution space distribution can be found.

Some authors consider a large number of stages and speak of interactive methods (López Jaimes & Coello Coello, 2013). These methods collect user preferences and give the possibility for the user to be active during the solution search process. The user has the opportunity to learn about the problem while exploring the available solutions (Miettinen, et al., 2008). This idea has been exploited in scalarization-based methods, such as (Linder , et al., 2012) or (Monz, et al., 2008). It has also been used with an evolutionary algorithm (Sinha, et al., 2014) (Bechikh, et al., 2015). Both approaches have some drawbacks, depending on the size and complexity of the problem (nature and number of objective functions). Three types of specifying preference information were identified by (Miettinen, et al., 2008): trade-off information, reference points and classification of objective functions. Our approach follows the idea of trade-off information by delimiting an area in the search space. But it does not belong to the interactive method class, in the sense that it only interacts once with the user. Consequently, our proposal for reducing optimization computation is a two-stage EA-based approach.

## 5.2 A proposal to reduce computation time for the CPPC problem: CFB-EA+

Within previous non-interactive methods, the partitioning of criteria or solution spaces and their affectation to a different optimization stage means that before launching the optimization, the user needs to have a priori preferences about the importance of the criteria, as well as the knowledge of the solution space areas to investigate. As CFB-EA allows all criteria to be considered over the whole solution space, a key idea of the proposal is to avoid the need for these a priori blind preferences by showing the user a first Pareto that consider the whole problem. Then, given this Pareto knowledge, the user can decide on how to compromise between criteria and solution space.

### 5.2.1 Description of CFB-EA+

Our proposal, called CFB-EA+, mixes both solution space and criteria partitions but only when knowing a little about the solution space. The idea is to launch a first optimization stage on the whole solutions space while taking into account all criteria, and to stop this process once a first 'raw' Pareto front can be identified at a time called switching time (ST). This Pareto result is shown to the customer in order for him/her, knowing the first general tendency of optimal compromises, to select a multi-criteria restricted area matching his/her criteria expectations.

We have seen, during the definition of the optimization model of the CPPC problem that each optimization criterion was modeled as a numerical constraint that sums CSP variables (cost attributes or operation durations). Thus, for the second optimization stage, these numerical constraints are bounded with maximum values that correspond to the restricted area that fits customer expectations. These numerical constraints are added to the CPPC model and the three CFB-EA sub-steps that are subject to constraint filtering (initialization of individual population, individual crossover and individual mutation), all respect these criteria bounds. Therefore, the proposed second stage of the CFB-EA generates individuals that respect customer criteria restriction expectations.

This two-stage optimization process is illustrated in Figure 31. On the left side, we can see the conventional single stage optimization that takes a long time. In the center, at the switching time, a first raw Pareto (resulting from the first stage optimization on the whole solution space) that allows capturing customer preferences (here the restricted area matching customer expectations is a maximum cycle time) is presented. On the right side, the final Pareto on the restricted area is shown.

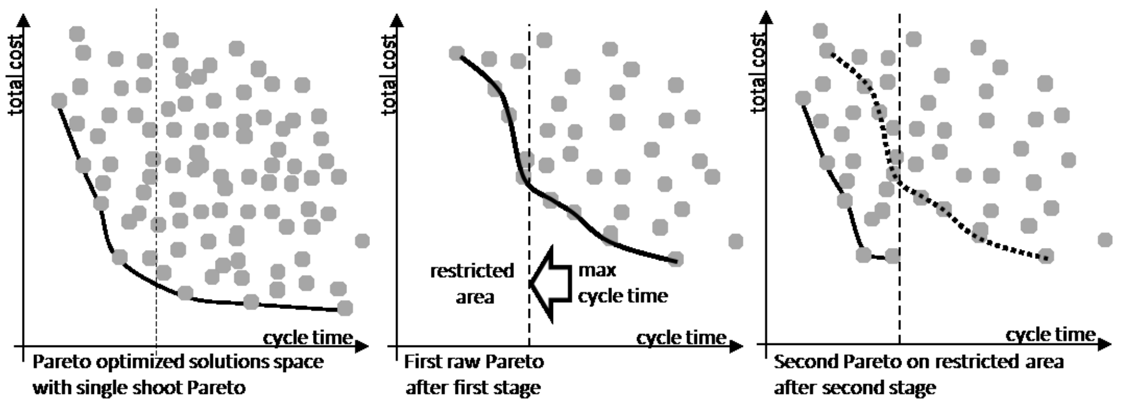


*Figure 31- Proposed two-stage optimization process*

The CFB-EA algorithm described in Figure 8 (section 2.2.5) is updated as follows: After the stopping criterion test step: (1) a switching time test is added, if the user is satisfied with the raw Pareto and has decided on a restricted area to investigate, (2) criteria constraints are inputted, (3) archive individuals that do not respect criteria constraints are removed, (4) criteria constraints are added to the filtering engine. The resulting CFBEA+ flowchart is shown in Figure 32, with new steps in bold dark grey. The problem of the tuning of the switching time will be addressed in the evaluation section at the end of Section (5.4.3).

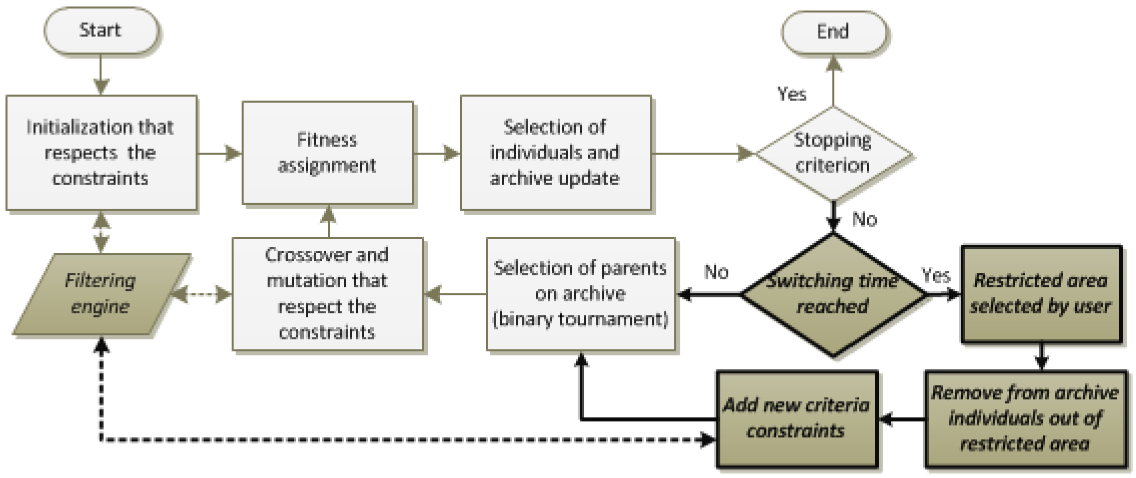

Start
End
Initialization that respects the constraints
Fitness assignment
Selection of individuals and archive update
Stopping criterion
Yes
No
Filtering engine
Crossover and mutation that respect the constraints
Selection of parents on archive (binary tournament)
No
Switching time reached
Yes
Restricted area selected by user
Add new criteria constraints
Remove from archive individuals out of restricted area

*Figure 32- CFB-EA+ algorithm*

### 5.2.2 Interests and limits of CFB-EA+

The three following interests can be underlined. Firstly, our idea globally corresponds to some kind of criteria ordering, but in this case, ordering is declared once a global tendency of solutions distribution with respect to criteria is known by the user. This means that CFB-EA+ avoids the 'blind choice' about criteria of other two-stage optimization approaches that rely on a priori criteria ranking. Secondly, the first optimization stage is identical to CFB-EA, but the second one does not restart from scratch. In fact, it benefits from the first-stage individuals that respect the restricted area and that are systematically included in the initial population of the second optimization stage. Thus there is no loss of computation time between the two optimization stages. Thirdly, we have seen in Section 1.2.2 that, depending on the quantity of elementary requirements, the size of the problem to optimize can vary. Furthermore, according to the content of these elementary requirements (high or low product performance, for example, that essentially drives selling price and cycle time), the requirement-respecting solution space that needs to be optimized can be clearly located in a different space area as shown in Figure 4. As CFB-EA+ provides a general tendency before asking for criteria preference, any kind of customer requirements can be easily handled even if the interesting solution locations are very different.

A limitation could be discussed concerns backtracking. As constraint filtering is not powerful or strong enough, when the restricted area corresponds to too hard constraints, the constraint filtering process of the CFB-EA specific sub-steps (individual crossover and individual mutation) can reach inconsistencies. By this, we mean that no solution remains for a configuration CSP variable during crossover or mutation. Thus, some backtrack processes are necessary to repair the solution. But if in CFB-EA, the backtrack is not frequent at all, with CFB-EA+ and criteria constraints that are stronger, the optimization process can spend a long time on backtracking. The experimentations of Section 5.4 will show that backtrack is from 10 to 40 times more frequent with CFB-EA+.

### 5.2.3 Tuning CFB-EA+ parameters

In order to use CFB-EA+, some recommendations relevant to the tuning of the two parameters must be proposed. We first define and discuss the tuning parameters, then we propose some recommendations that will be validated according to the result of the experimentation section. The two parameters are the "switching time" between the two-optimization stage and the "size of the restricted area" matching customer expectations.

About the switching time:

- If it is too early, the first optimization stage might be unable to provide a rough Pareto sufficiently detailed to allow capturing a suitable restricted area matching customer expectations.
- If it is too long, the first optimization stage will be very close to the optimal value and the computation time reduction will be rather low.

About the size of the restricted area matching customer expectations:

- If is too large, the second optimization stage will have a large solution space to investigate and, as before, the computation time reduction will be rather low.
- If is too small, the evolutionary operators may fall in a local optimal area missing the global optimal one.

These two tuning parameters are more or less linked. Let us first consider the size of the restricted area. Once the switching time is reached, the raw Pareto front is shown to the user and the idea is to ask him/her, knowing current criteria values, about a solution tendency. By solution tendency, we mean for two criteria to indicate either a preference on a single criterion or a compromise between the two criteria, this provides three criteria tendencies for reducing the search space (for example on figure 33: cycle time < ct1, total cost < tc1, cycle time < ct2 and total cost < tc2). Thus, as a suggestion, the raw Pareto front could be divided in to three parts containing a same number of individuals (1/4 of Pareto front that gathers 3 or 4 points each in figure 33). This is an order of magnitude suggestion and, of course, it is possible to deviate from this solution according to customer expectations.

Once this restricted area is defined, in order to avoid the local optimal problem, it is necessary to check if the quantity of individuals existing in this area ($Q_{tt}$ in area) at the switching time is sufficient. This value will be quantified according to further experimentations. If it is not, this means that the switching time is too early and that the first optimization stage must be continued or that the size of the constrained area should be increased.

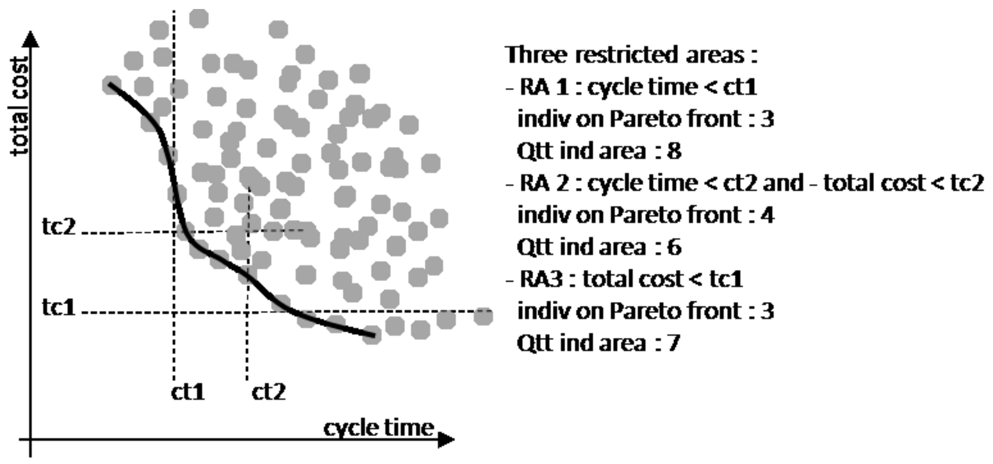


*Figure 33- Different restricted areas for second optimization stage*

### 5.3 Experimental Plan

In this section, we first present the correction of evaluation metric needed to compare CFB-EA and CFB-EA+. Then we describe our experimentation plan.

#### 5.3.1 Performance evaluation for experiments with CFB-EA+

Hypervolume computed during first stage corresponds to the area covered in the whole search space. When evaluating CFB-EA+, we must correct the value of this hypervolume with respect to the switching time (ST) between the two optimization stages and the associated restricted area. This means that once the two-stage optimization is over, when plotting the HV versus computation time:

- At the before switching time, we consider a corrected hypervolume as shown in the center of Figure 34, which means that we discard all individuals outside the restricted area when computing HV,
- after switching time, as the constraints corresponding to the restricted area are respected by any individual, HV is unchanged, as shown on the right of Figure 34.

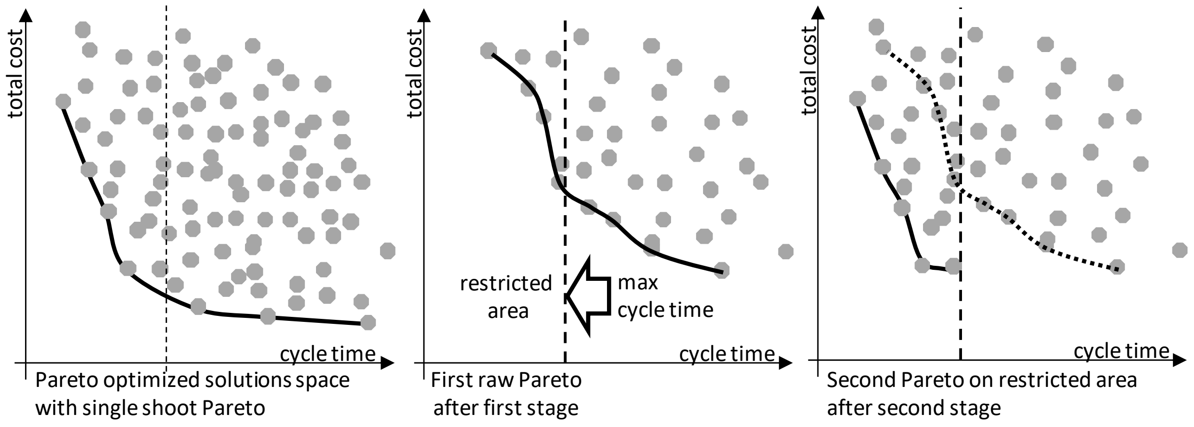

total cost
cycle time
Pareto optimized solutions space with single shoot Pareto
total cost
restricted area
max cycle time
cycle time
First raw Pareto after first stage
total cost
cycle time
Second Pareto on restricted area after second stage

*Figure 34- Hypervolume computation for CFBEA+*

In order to be able to compare CFB-EA with CFB-EA+ on a given restricted area, the hypervolume relevant to CFB-EA will be computed with the CFB-EA+ mode used before switching time, meaning that HV computation will not consider individuals outside the restricted area. Thus, before switching time the HV time evolution is formally similar for both algorithms, then after switching time a different behavior should be observable, as shown in Figure 35.

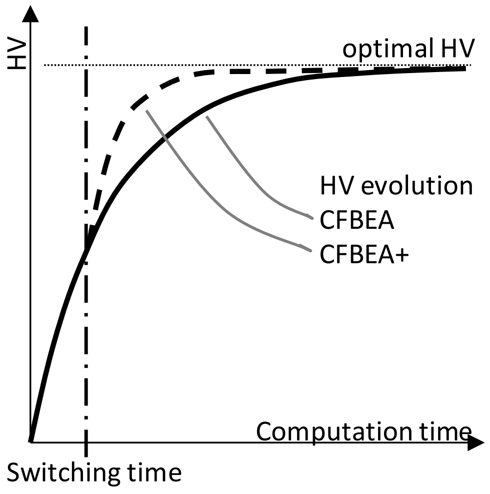


*Figure 35- Comparing HV evolution of CFB-EA+ and CFB-EA*

The main experimentations in order to compare CFB-EA and CFB-EA+ will be achieved with the reference case (platform model as described in section 4.3). Then a larger problem (intermediate size case) will be considered to show the scalability of the proposal.

### 5.3.2 Experimentation plan

The experimental plan showed in this part corresponds to the one published in (Pitiot, et al., 2019). This publication was achieved with an older version of CFB-EA than the one used in chapter 4. As it does not benefit from the last improvements made during the end of thesis, time consumption is higher than the ones presented in chapter 4. Nevertheless, comparison of CFB-EA and CFB-EA+ with the same version and on the same models is relevant.

CFB-EA reference results are obtained as follows:

(1) the CFB-EA algorithm is launched 5 times with a time bound of 9000 seconds,

(2) the average values of the first ($HV_{first}$) and final ($HV_{final}$) hypervolume are computed,

(3) the average time to reach $HV_{final}$ is also noted. It might be below 9000 seconds.

It is important to note that these previous values consider the whole solution space. Then, considering the Pareto front associated with the previous $HV_{final}$, and in order to achieve comparisons with CFB-EA+:

(4) three restricted areas (RA) illustrated in figure 36 are defined as:

- RA1: cycle time < $ct_{max}$, $ct_{max}$ defined with 25% individuals on Pareto front,
- RA2: cycle time < $ct_{med}$ & total cost < $tc_{med}$ defined with 25% individuals on Pareto front
- RA3: total cost < $tc_{max}$, $tc_{max}$ defined with 25% individuals on Pareto front.

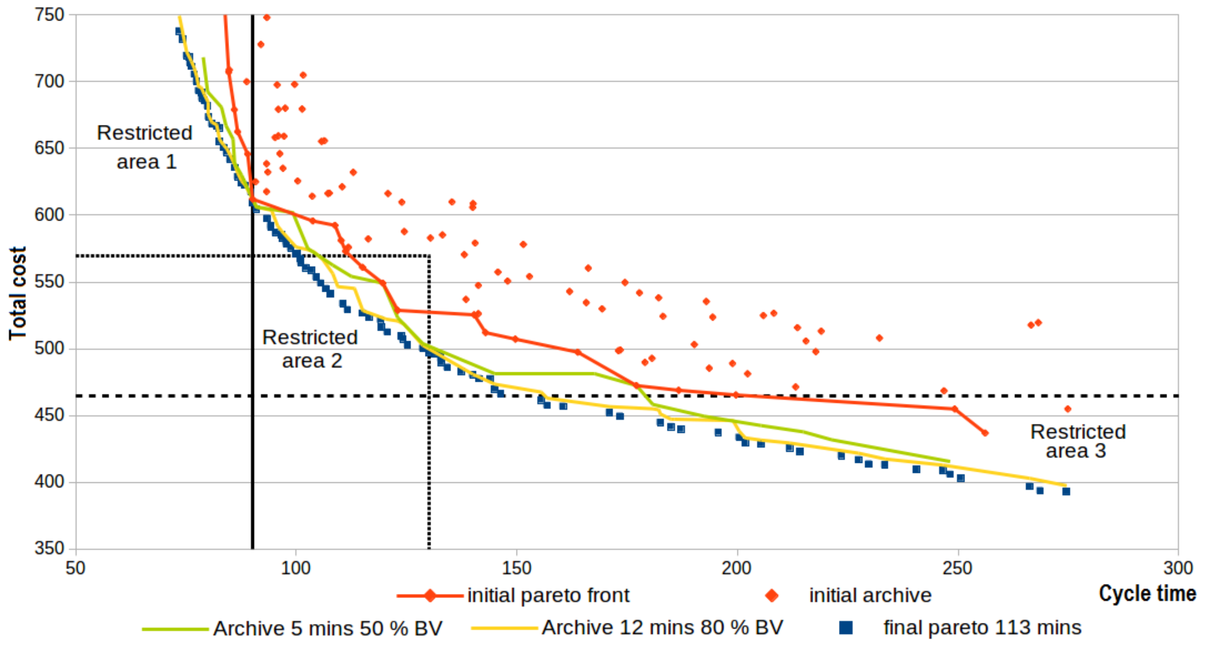


*Figure 36- Four restricted areas for CFB-EA+*

(5) and for each restricted area:

- the corrected area hypervolume of CFB-EA (noted "Hypervolume best value BV" or "BV" in the results tables) is computed,
- the time to reach 95%, 99%, 99.9% and 100% of previous CFB-EA "BV" is also computed.

For each of these times and HV values, an average and a relative standard deviation (RSD) are provided. It can be seen in Figure 36 that in 12 minutes the Pareto front gives a good idea of possible potential compromises.

In order to avoid absolute values for switching times, the analysis is conducted with three switching times related to different levels of Hypervolume for each previously defined restricted area. These three switching times correspond to a HV value of 70, 80 and 90% of ($HV_{final}$ - $HV_{first}$). They correspond roughly to 466, 800 and 1160 seconds. Each of these times is associated with a number of individuals for each restricted area (noted $Q_{tt}$ in the tables of results), that have been generated during the first computation and that are in the restricted area, including, of course, those that belong to the Pareto front.

For each couple (restricted area, switching time), CFB-EA+ is launched five times with the same time bound of 9000 s, and the following values are recorded:

- average of the previous number of individuals ($Q_{tt}$),

- the times when the hypervolume reaches 95%, 99%, 99.9% and 100% of the hypervolume best value (BV) of CFB-EA,
- the final or best hypervolume,

As before, both the average and the RSD are provided for these times and HV values. These results are compared to the results provided by CFB-EA on the same restricted area with a gap percentage, 100 * [value (CFB-EA) - value (CFB-EA+)] / value (CFB-EA).

Having conducted CFB-EA optimization with various problem sizes, we came to the conclusion that the adequate size of the archive is 100, adequate population size is 150, appropriate crossover probability for individual selection is 0.8, and mutation probabilities for individual and gene selections should be set respectively at 0.5 and 0.1.

## 5.4 Results

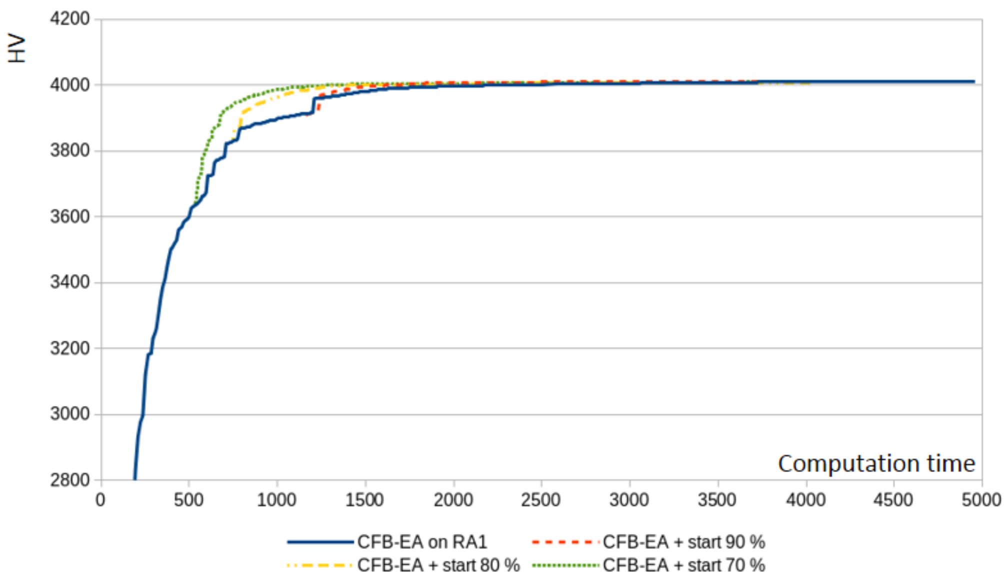
In the following section, we present the associated results, discuss them and propose some recommendations for using and tuning CFB-EA+. Then with a larger model, we discuss scalability issues.

### 5.4.1 CFB-EA and CFB-EA+ results comparison and discussions

The hypervolume evolutions are shown and computations times are compared for the three restricted areas.

#### 5.4.1.1 Restricted Area 1, cycle time constraint

Figure 37 shows the hypervolume evolution on a graph with a table that details the computation times.



| Restricted area 1 | CFBEA | | CFBEA+ 70% | | | CFBEA+ 80% | | | CFBEA+ 90% | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 25% max time | | | Switch. Time : 466 - $Q_{tt}$ : 36,4 | | | Switch. time : 800 - $Q_{tt}$ : 65 | | | Switch. time : 1100 - $Q_{tt}$ : 100 | | |
| Time to reach : | Average | RSD (%) | Average | , | % gap | Average | RSD | % gap | Average | RSD | % gap |
| - 95% of BV CFBEA | 709,14 | 4,60 | 607,20 | 3,71 | 14,38 | | | | | | |
| - 99% of BV CFBEA | 1360,91 | 1,90 | 874,94 | 2,29 | 35,71 | 1040,56 | 2,29 | 23,54 | 1259,46 | 1,24 | 7,45 |
| - 99,9% of BV CFBEA | 2972,05 | 9,47 | 2164,38 | 27,78 | 27,18 | 2219,21 | 23,82 | 25,33 | 1843,17 | 4,98 | 37,98 |
| - 100% of BV CFBEA | 4958,03 | 30,80 | 4217,23 | 43,80 | 14,94 | 4047,02 | 49,22 | 18,37 | 4369,32 | 22,14 | 11,87 |
| Hypervolume best value BV | 4009,96 | 0,00 | 4009,96 | 0,00 | 0,00 | 4009,96 | 0,00 | 0,00 | 4009,96 | 0,00 | 0,00 |

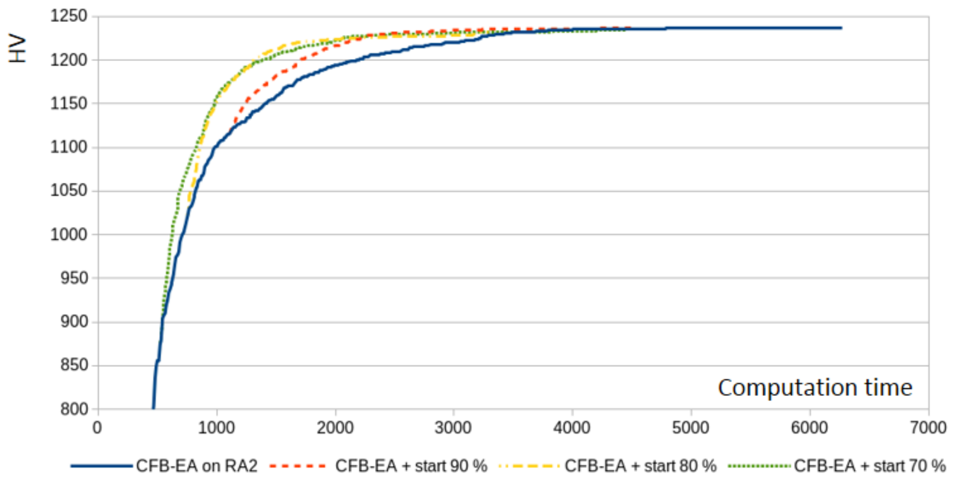
*Figure 37- Comparison CFBEA+ with CFBEA on restricted area 1*

For any switching time and any table times (95%, 99%, 99.9% and 100% and BV) CFB-EA+ computation times are always lower, with a gap between 6% and 43%. For CFB-EA+, as switching time 80% (800 s.) and 90% (1160 s.) are higher than CFB-EA time to reach 95% of BV CFB-EA (around 530 s.), there is no computation time comparison. It can be noticed that each of the five runs reaches the hypervolume best value (BV 4009.96 with RSD = 0). Furthermore, even if CFB-EA+ takes more time to build individuals (CFB-EA+ backtrack rate by an individual is around 4.25%, while CFB-EA backtrack rate is close to 0.4%), it reaches BV more quickly.

#### 5.4.1.2 Restricted Area 2, cycle time and total cost constraint

Results are shown in figure 38. These results show some interesting behavior. This restricted area is the most highly constrained one. The backtrack rate by an individual is around 17% (more than 40 times more than CFB-EA). It also has the lowest quantity of individuals from the first stage (respectively 20.4, 51.8 and 88.4). That leads to, with early switching times (70% and 80%) CFB-EA+ is unable to reach 99.9% and 100% of CFB-EA best value. However, those final values are very close to the hypervolume best value and they are reached very early. Moreover, all other values show lower computation times with gaps between 14% and 37%. Even if individuals are more difficult to obtain, due to backtracking, CFB-EA+ outperform

CFB-EA. The key point would be to avoid the lack of diversity in the initial population that leads to local optimal for some cases of the earliest switching time.

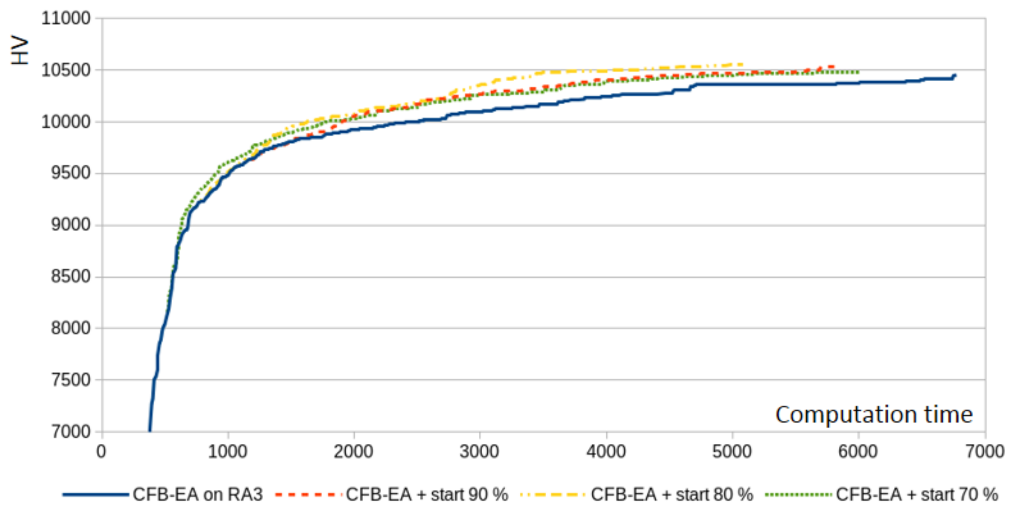


| Restricted area 2 | CFBEA | | CFBEA+ 70% | | | CFBEA+ 80% | | | CFBEA+ 90% | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 25% max time and max cost | | | Switch. time : 466 - $Q_{tt}$ : 20,4 | | | Switch. time : 800 - $Q_{tt}$ : 51,8 | | | Switch. time : 1160 - $Q_{tt}$ : 88,4 | | |
| Time to reach : | Average | RSD (%) | Average | RSD | % gap | Average | RSD | % gap | Average | RSD | % gap |
| - 95% of BV CFBEA | 1660,49 | 2,35 | 1106,09 | 2,27 | 33,39 | 1118,49 | 2,28 | 32,64 | 1427,11 | 1,91 | 14,06 |
| - 99% of BV CFBEA | 3212,22 | 2,38 | 2051,04 | 6,10 | 36,15 | 2012,20 | 4,28 | 37,36 | 2177,17 | 4,68 | 32,22 |
| - 99,9% of BV CFBEA | 4039,83 | 14,58 | | | | | | | 3242,81 | 18,91 | 19,73 |
| - 100% of BV CFBEA | 6272,75 | 24,75 | | | | | | | 4515,45 | 31,52 | 28,01 |
| Hypervolume best value BV | 1236,29 | 0,00 | 1234,43 | 0,45 | 0,15 | 1232,57 | 0,60 | 0,30 | 1236,29 | 0,00 | 0,00 |

*Figure 38- Comparison CFBEA+ with CFBEA on restricted area 2*

### 5.4.1.3 Restricted area 3, total cost constraint

Results are shown in figure 39. For this restricted area, the tendency is similar. The backtrack rate by an individual is around 1,7%. The CFBEA+ computation times are still always better than CFB-EA with values between 11% and 49%. The highest decreases are obtained with the intermediate switching time (80%) that also always provides the best hypervolume value (10552.70).



| Restricted area 3 | CFBEA | | CFBEA+ 70% | | | CFBEA+ 80% | | | CFBEA+ 90% | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 25% max cost | | | Switch. time : 466 - $Q_{tt}$ : 30,2 | | | Switch. time : 800 - $Q_{tt}$ : 66,8 | | | Switch. time : 1160 - $Q_{tt}$ : 100 | | |
| Time to reach : | Average | RSD (%) | Average | RSD | % gap | Average | RSD | % gap | Average | RSD | % gap |
| - 95% of BV CFBEA | 2044,01 | 4,46 | 1625,06 | 2,04 | 20,50 | 1466,13 | 1,55 | 28,27 | 1816,75 | 1,31 | 11,12 |
| - 99% of BV CFBEA | 4711,54 | 23,59 | 3668,07 | 7,01 | 22,15 | 2980,75 | 6,28 | 36,74 | 3628,91 | 6,82 | 22,98 |
| - 99,9% of BV CFBEA | 6748,95 | 28,50 | 4832,31 | 6,84 | 28,40 | 3434,27 | 10,69 | 49,11 | 4398,08 | 13,11 | 34,83 |
| - 100% of BV CFBEA | 6772,20 | 28,77 | 5019,18 | 9,31 | 25,89 | 3444,59 | 10,85 | 49,14 | 4605,86 | 14,73 | 31,99 |
| Hypervolume best value BV | 10454,90 | 2,17 | 10481,00 | 2,05 | -0,25 | 10552,70 | 0,00 | -0,94 | 10539,20 | 0,26 | 0,81 |

*Figure 39- Comparison CFBEA+ with CFBEA on restricted area 3*

### 5.4.2 <u>Global results comparison</u>

Table 4 gathers all computation time reductions as a percentage decrease with respect to various combinations of restricted areas and switching times.

When globally comparing all computation times, we obtain 30 times gaps: 10 with RA1, 8 with RA2 and 12 with RA3. The average of these 30 gaps is 27.05%, meaning that CFB-EA+ allows a significant reduction in computation time.

When comparing reductions with respect to restricted areas, the average gap of RA1 is 21.68%, while RA2 is 29.19% and RA3 is 30.09%, meaning that CFB-EA+ computation time reductions are a little higher when restricted areas are more constrained by total cost.

When considering only the time required to obtain the best hypervolume value (100% of BV CFB-EA), CFB-EA+ is always better than CFB-EA, with an average gap around 26% (excluding restricted area 2 with 70% and 80% switching time). This result, close to the global one (27.05), shows that significant computation time reductions are obtained with CFB-EA+ without any reduction of optimality level of the solutions.

When taking into account the different switching times, average gaps are as follows: CFB-EA+ 70%: 25.87%, CFB-EA+ 80%: 33.39% and CFB-EA+ 90%: 22.93%. This means that the highest computation time reductions are provided with the intermediate switching time associated with a hypervolume equal to 80% of ($HV_{final}$ –$HV_{first}$).

As previously mentioned, the quantity of individuals respecting the restricted area at the switching time (noted “$Q_{tt}$” in the tables of results) at the beginning of the second CFB-EA+ optimization stage must be considered as a switching time condition to avoid local optimums. Considering Table 4, we suggest a minimum quantity of 60 for all restricted areas. This quantity of individuals will be the basis for our tuning recommendation of the next section. Of course, this quantity is roughly related to the size of the archive, which is 100 in previous experiments.

| Computation times comparisons | CFBEA + 70% Switch. time: 325 s | CFBEA + 80% Switch. time: 800 s | CFBEA + 90% Switch. time: 1100 s | % Gap average wrt restricted area |
|---|---|---|---|---|
| Restricted area 1: 25% max time | Qtt: 36,4 | Qtt: 65 | Qtt: 100 | |
| Time to reach: | % gap | % gap | % gap | |
| 95% of BV CFBEA | 14,38 | | | |
| 99% of BV CFBEA | 35,71 | 23,54 | 7,45 | All times to reach |
| 99,9% of BV CFBEA | 27,18 | 25,33 | 37,98 | 21,68 |
| 100% of BV CFBEA | 14,94 | 18,37 | 11,87 | 15,06 |
| % Gap average (Rest. area and Swithching time) | 23.05 | 22.41 | 19,10 | |
| Restricted area 2: 25% max time and max cost | Qtt: 20,4 | Qtt: 51,8 | Qtt: 88,4 | |
| Time to reach: | % gap | % gap | % gap | |
| 95% of BV CFBEA | 33,39 | 32,64 | 14,06 | |
| 99% of BV CFBEA | 36,15 | 37,36 | 32,22 | All times to reach |
| 99,9% of BV CFBEA | | | 19,73 | 29,19 |
| 100% of BV CFBEA | | | 28,01 | 28,01 |
| % Gap average (Rest. area and Swithching time) | 34,77 | 35,00 | 23,51 | |
| Restricted area 3: 25% max cost | Qtt: 30,2 | Qtt: 66,8 | Qtt: 100 | |
| Time to reach: | % gap | % gap | % gap | |
| 95% of BV CFBEA | 20,50 | 28,27 | 11,12 | |
| 99% of BV CFBEA | 22,15 | 36,74 | 22,98 | All times to reach |
| 99,9% of BV CFBEA | 28,40 | 49,11 | 34,83 | 30,09 |
| 100% of BV CFBEA | 25,89 | 49,14 | 31,99 | 35,67 |
| % Gap average (Rest. area and Swithching time) | 24,23 | 40,81 | 25,23 | |
| % Gap average wrt swithching time | 25,87 | 33,39 | 22,93 | % Gap avearge global 27,05 |

*Table 4- Time reductions with respect to restricted areas and switching times.*

### 5.4.3 CFB-EA+ tuning recommendation for switching time and restricted area

The recommendation process works in the following way: once the first CFB-EA+ optimization stage is launched on the whole solution space, at each loop, the possibility for a CFB-EA+ second stage initialization is checked as follows:

- the individuals of the Pareto front are analyzed, and the constraints associated with each restricted area are computed, with respect to a 25% minimizing of each criterion and 25% of central compromise of the two criteria,
- the quantity of individuals included in each restricted area since the beginning of the CFB-EA+ stage is calculated,

- for each restricted area:
    - if this quantity of individuals is smaller, the CFB-EA+ first stage carries on,
    - if this quantity of individuals is larger than 60 (given the archive is around 100 in our case), the process proposes to the user to launch the second CFB-EA+ optimization stage,
    - if the user is not satisfied by the proposed restricted area, CFB-EA first stage carries on,
    - if the user is satisfied by the proposed restricted areas, he can either validate the proposed restricted area or slightly modify it, as long as the minimum quantity is respected.

This recommendation will be used for optimizing the larger problem of the next section. The initial CFB-EA+ algorithm shown in Figure 32 Section 5.2.1 is updated and gives the final CFB-EA+ algorithm that works as illustrated in Figure 40.

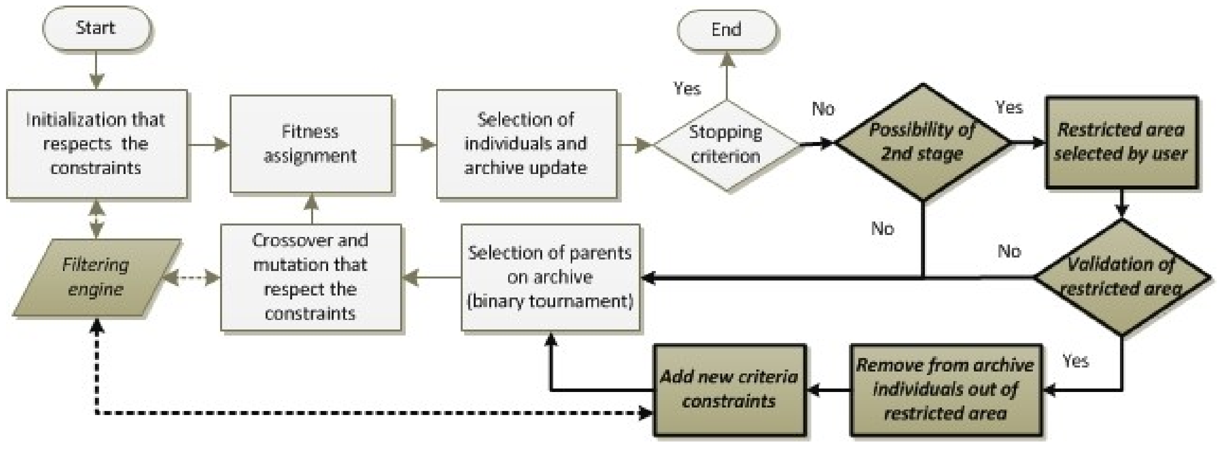
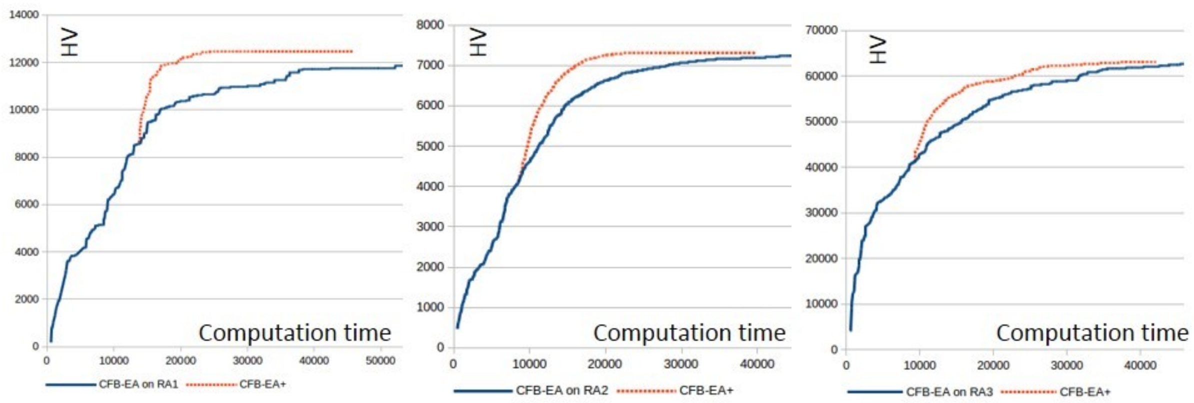

Start
End
Initialization that respects the constraints
Fitness assignment
Selection of individuals and archive update
Yes
Stopping criterion
No
Possibility of 2nd stage
Yes
Restricted area selected by user
Filtering engine
Crossover and mutation that respect the constraints
Selection of parents on archive (binary tournament)
No
No
Validation of restricted area
Yes
Add new criteria constraints
Remove from archive individuals out of restricted area

*Figure 40- CFB-EA+ algorithm including the tuning process*

### 5.4.4 Experiments with a much larger model and scalability issues

In order to deal with scalability issues, we consider now the intermediate case with 60 variables. The archive size is larger, 150 (instead of 100), and the number of individuals that triggered the CFB-EA+ second stage initialization possibility is set to 100 (instead of 60) for all restricted area. The resulting switching times are, for RA1: 13736, for RA2: 7835 and for RA3: 8901. Figure 41 shows the hypervolume evolution for each restricted area with tables showing computer time reductions.



| | CFBEA | CFBEA+ Restrict. area 1 | | CFBEA | CFBEA+ Restrict. area 2 | | CFBEA | CFBEA+ Restrict. area 3 | |
|---|---|---|---|---|---|---|---|---|---|
| | | ST : 13736 - $Q_{tt}$: 100 | | | ST : 7835 - $Q_{tt}$ : 100 | | | ST : 8901 - $Q_{tt}$ : 100 | |
| Time to reach : | Average | Average | % gap | Average | Average | % gap | Average | Average | % gap |
| - 95% of BV CFBEA | 37731 | 16961 | 55,05 | 27378 | 15834 | 42,17 | 31590 | 22729 | 28,05 |
| - 99% of BV CFBEA | 82766 | 20002 | 75,83 | 46443 | 19899 | 57,15 | 43844 | 31688 | 27,73 |
| - 99,9% of BV CFBEA | 86144 | 21723 | 74,78 | 77578 | 25719 | 66,85 | 73122 | 37573 | 48,62 |
| - 100% of BV CFBEA | 86477 | 21760 | 74,84 | 82119 | 36015 | 56,14 | 74257 | 39127 | 47,31 |
| Hypervol. best value BV | 12283,80 | 12466,10 | 1,48 | 7318,55 | 7318,94 | 0,01 | 63120,20 | 63190,70 | 0,11 |

*Figure 41- Comparison CFB-EA+ / CFB-EA with a larger model*

These curves and time reductions show that CFB-EA+ still works fine with larger problems. Furthermore, it can be noticed that the computation time reductions are globally larger with this 60-variables model (around 54%) compared to the 30-variables model described in the previous section (around 27%). This leads us to the conclusion that the interest of CFB-EA + is greater when the amount of elementary requirements provided by the user is small. Let us imagine, although this was not the case for previous experiments, that:

- the real problem has a model size of 80 variables,
- that results provided in section 5.4.1 correspond with 50 elementary requirements,
- that results provided in section 5.4.4 correspond with 20 elementary requirements.

Results show, that for a given size problem of 80, better time reductions are obtained with 20 elementary requirements compared to 50. Thus, for our interactive configuration problem, if requirements are very detailed and cover almost all configuration variables, our recommendation would be to just use the CFB-EA single-stage approach.

### 5.5 Conclusion on CFB-EA+ evaluation:

The key idea of CFB-EA+ is to avoid processing the whole solution space, but also to avoid “a priori” ordering of criteria and problem variables. The idea is to quickly compute a first Pareto front and, according to the knowledge of this result, to suggest some restricted areas to the user which will be subject to further enforced investigations in a second stage.

With regard to computation time reductions, a low-constrained model of 30 configurable variables was used for experiments with various switching times and restricted areas. Without any parameter classification, a decrease in the order of magnitude of 30% of computation time can be expected. It has also been noted that this improvement is obtained without any decrease of the optimality level of the solutions.

For the tuning of CFB-EA+ parameters, previous experiments have confirmed that the largest reductions in computation time were obtained with an intermediate switching time. But as a switching time associated with a small restricted area could lead to an over-constrained problem or a lack of diversity in the initial population, the quantity of individuals in a given restricted area has been preferred as a necessary tuning parameter. Given these results, we have proposed an updated version of CFB-EA+ that enables to suggest to the user switching times associated with quantity of individuals in selected area of interest.

A model gathering 60 variables with characteristics similar to the previous one was also considered. Experiments confirmed previous computation time reductions and show even better results for a larger model.

These conclusions clearly show the interests of the CFB-EA+ approach in reducing optimization computation times. Furthermore, considering company expectations, and especially in B2C configuration situations, the possibility of showing a first Pareto Front in less than 15 min, and thus providing an idea of the solution space distribution will undoubtedly serve to keep the customer's attention focused on the configuration possibilities. By viewing this Pareto, the customer can express a well-founded expectation of his/her criteria preferences and therefore feel greater satisfaction. For B2B situations, the contribution is a little different, as the customer usually has to wait for an optimal solution, which takes quite a long time.

Here, even if this computation time is significantly reduced, it is possible to provide the customer with partial solution-space tendencies in a relatively short time, thus limiting the search for optimal space regions.

# 6. Conclusions and future works

## 6.1 Conclusion

This PhD focuses on modeling and optimizing issues for the O-CPPC problem as well as evaluating optimization approach (CFB-EA). It was motivated by strong expectations of the industrial issues: the need of powerful handling and optimization tools for a difficult decision aiding problem (large, multiobjective, combinatorial and constrained optimization) as well as the need of proper evaluation of optimization approach. The domains of O-CPPC problem and associated definitions, the frameworks and tools used (CSP and EA) were presented as well as a state-of-art of them. Responses to the three research questions were proposed in this work:

<u>QR1: "Is it possible to propose a generic model of the CPPC problem that can avoid case dependency when evaluating and comparing optimization methods?"</u>

A generic model and the associated generator have been proposed and discussed. The generic model gathers different points of view of product modeling (physical, functional or mixed) that could be found in various industrial cases. It proposes a structured decomposition of CPPC models with definition of:

- architectures (platform, modular and integrated),
- physical-functional modules,
- Product Configuration Evaluation Patterns (5 patterns),
- Product Configuration Constraints Patterns (4 patterns),
- Product Evaluation Constraints Patterns (3 patterns),
- A Process operation pattern,
- Process evaluation pattern that links workload, resource, cost and duration of tasks,
- Rules for constraint distribution between decision variables in each domain (product, process or coupling).

These numerous elements must allow to capture a large diversity of existing configuration problems. Such generic modelling clearly avoids problems of case-dependency or random generation of models that suffer most of existing publications.

QR2: How sensitive is CFB-EA optimization method, with respect to each key characteristic of the generic model of the CPPC problem?

Given generic model described, a first version of a benchmark for O-CPPC optimization problems was proposed with eight cases. It allowed us to evaluate three key characteristics of CPPC problem: product architecture, size and constraints density.

The study confirms abilities of CFB-EA in relation with size and constraint density. Time consumption are relevant for the addressed decision aiding process. Experiments showed some interesting behaviors in relation to the Product architecture even if its impact is less relevant than size or constraint density.

Formally, strong constrained cases (high constraint density level or concentrated constraints distribution as in platform case) lead to an easier optimization, whereas, low constrained cases (low constraint density level or constraints well spread all over the model as in modular case) are harder to optimize finely.

In all cases, CFB-EA reaches a near-optimal (99% of final performance) Pareto front very quickly (less than 20% of the time to reach the best solution or less than 2 hours for the largest considered problem).

QR3: Is it possible to reduce the computation times of CFB-EA and other conventional EA approaches?

A two-step approach called CFB-EA+ was proposed and evaluated. It takes benefits from CFB-EA initial fast improvement to propose a first view of solution space ("raw" Pareto). It takes benefit to user's preference collection as in interactive methods but without asking him several times. It also avoids the blind choice of partition of criteria or solution spaces of "a priori" approaches.

CFB-EA+ shows a time consumption decrease in the order of magnitude of 30% in 30 variables model and up to 50% in larger cases (60 variables). Those improvements are obtained with similar or better performance than with CFB-EA.

According to the analyze of parameters tuning, the diversity of initial solutions in second phase is the key point to obtain the most significant time reduction without falling in a local optimum. Therefore, we have proposed an updated version of CFB-EA+ that enables to suggest to the user switching times associated with quantity of individuals in selected area of interest.

## 6.2 Futures works

Presented works constitute a strong base for analysis, modeling and optimization of CPPC problem. For each research question, following perspectives could be investigated:

QR1: Generic model of the CPPC problem

The generic model proposed gathers various patterns coming from our experience on industrial projects. It could be completed with some other patterns that could be relevant to: specific industrial situations. In a similar way, another key issue that represents a popular research theme could be to integrate other evaluation criteria linked to environmental impact as carbon foot print or to product/process quality.

A problem for optimization researchers is that real cases models are most of the time hidden by the privacy policy of companies. Thanks to the model generator and procedure, companies interested on the subject could build a kind of abstract model of their own product/process and add it to the benchmark to know which optimization seems to be the most accurate. In a same way, it is also necessary to use a standard format to describe cases. At the present time, they are described in the input format of our filtering engine. Based on XML format, XCSP3 proposed by (Boussemart , et al., 2016) seems to be a good candidate disconnected from economic issues of configurator software editors. Furthermore, a competition is annually organized to solve COP (constraints optimization problems) models formatted with XCSP3 problem. Once achieved, the last issue is to publish this CPPC model generator on the web in order to capture applied companies cases.

QR2: Evaluation of CFB-EA characteristics on CPPC benchmark

The experiments plan showed in chapter 4 evaluates our last version of CFB-EA with a standard parameter setting. It has been selected according to a first small experiments plan achieved on reference case (platform basic case with medium constraints density and medium size). A finer experiments plan could be achieved for a better understanding of connections between evolutionary setting and properties of cases.

Once last improvements of CFB-EA published, another experimental study will be to evaluate performance of the new evolutionary operators with respect to the key characteristics of CPPC cases (model size, constraint density and architecture).

QR3: Improvements of CFB-EA and their evaluation

In the experimental plan, the evolutionary parameters setting was identical for both stages of CFB-EA+ and for CFB-EA in order to compare safely approaches. We have deduced that the key requirement for the second stage was a good diversity of initial population. Maybe, a specific tuning could be found for each stage: A setting that reinforces exploration and diversity in the first stage and another one that intensifies search during second stage. This last point can be delicate because if the first stage “misses” a potential set of different solutions, it would be difficult to design a setting that at the same time reinforces search around solutions already found and also searches for a missed one. Perhaps, a varying setting could achieve this difficult challenge.

Experiments of chapter 5 and those related in (Pitiot, et al., 2019) were not achieved with the very last version of CFB-EA and CFB-EA+. A new plan must be achieved to confirm gaps between CFB-EA and CFB-EA+ obtained with this last version.

Besides unpublished improvements already developed, other ways could be investigated. Coupling of CFB-EA with SLS (stochastic local search) is another way to avoid stagnation of optimization algorithm after the initial quick improvement.

Concerning other criterion added in the model, a delicate issue may be to express the constraints that characterize three-dimensional restricted areas.

The last work perspective concerns the possibility of using the CFB-EA+ key idea with other multiobjective evolutionary algorithms. For this last question, we have good confidence that the proposed ideas should work fine with any multiobjective and constrained evolutionary algorithms.

# 7. Bibliographical References

Aldanondo , M. & Vareilles, E., 2008. Configuration for Mass Customization: How to extend product configuration towards requirements and process configuration. *Journal of Intelligent Manufacturing ,* 19(5), pp. 521-535A.

Aldanondo, M., Hadj Hamou, K., Moynard, G. & Lamothe , J., 2003. Mass customization and configuration: Requirement analysis and constraint based modeling propositions. *Integrated Computer Aided Engineeirng ,* 10(2), pp. 177-189|.

Aldanondo, M., Vareilles, E., Dejel, M. & Galborit , P., 2008. *Towards an association of product configuration with production planning.* Patras, Greece, University of Patras .

Aldanondo, M., Vareilles, E. & Djefel, M., 2010. Towards an association of product configuration with product planning. *International Journal of Mass Customisation ,* 3(4), pp. 316-332.

Allen, J., 1983. Maintaining Knowledge about temporal intervals. In: *ACM 11.* Rochester : University of Rochester , pp. 123-154.

Amilhastre, J., 1999. *Représentation par automate d´ensemble de solutions de problèmes de satisfaction de contraintes. Doctoral Thesis ,* Montpellier: Université de Montpellier.

Amilhastre, J., Fargier, H. & Marquis, P., 2002. Consistency restoration and explanations in dynamics CSPs application to configuration. *Artificial Intelligence,* 135(1-2), pp. 199-234.

Anon., n.d.

Ascione, F. et al., 2016. Multistage and multiobjective optimization for energy retrofitting a devoloped hospital reference building. A new approach to assess cost optimality. *Applied Energy ,* Volume 174, pp. 37-68.

Bartak , R., Salido, M. & Rossi, F., 2010. Constraint satisfaction techniques in planning and scheduling. *Journal for Intelligent Manufacturing ,* 21(1), pp. 5-15.

Baxter, D., 2007. An engineering design kowledge reuse methodology using process modelling. *Research in Engineering Design,* 18(1), pp. 37-48.

Bechikh, S., Kessentini , M., Said, B. & Ghedira, K., 2015. Preference Incorporation in Evolutionary Multiobjective Optimization: A survey of the state of the arts. *Advances in Computer ,* Volume 98, pp. 141-207.

Blackstone, J., 2013. *APICS Dictionary.* Chicago : APICS.

Blot, A., Kessaci, M. & Jourdan , L., 2018. Survey and unification of local search techniques in metaheuristics for multiobjective combinatorial optimization. *Journal of Heuristics,* pp. 1-26.

Bonjour, E., 2008. *Contributions à l'instrumentation du métier d'architecte système : de l'architecture modulaire du produit àl'organisation du système de conception,* France: Université de Franche-Comté.

Bonjour, E., Deniaud, S., Dulmet, M. & Harmel , G., 2009. A fuzzy method for propagating functional architecture constraints to physical architecture. *Journal for Mechanical Design,* 131(6).

Bonjour, E. & Micaëlli, J. P., 2010. Design core competence diagnosis: a case from the automotive industry. *IEEE Transactions on Engineering Management ,* 57(2), pp. 323-337.

Boussemart , F., Lecoutre , C., Audemard, G. & Piette, C., 2016. *XCSP3: An Integrated Format for Benchmarking Combinatorial Constrained Problems. Technical Report,* Ithaca, New York: Cornell University Library.

Brown, D., 1998. Defining Configuring. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* Volume 12, pp. 301-305.

Campagna, D. & Formisano, A., 2013. Product and Production Process Modelig and Configuration. *Fundamenta Informaticae,* Volume 124.

Chen, G. et al., 2006. A framework for an automotive body assembly process design system. *Computer Aided Design,* 38(5), pp. 531-539.

Chen, L. & Lin, L., 2002. Optimization of product configuration design using functional requirements and constraints. *Research in Engineeirng Design ,* 13(3), pp. 167-182.

Chenouard, R., 2007. *Résolution par satisfaction de contraintes appliquée á l aide á la décision en conception architecturale. Doctoral Thesis,* Paris: ENSAM.

Coello, C., n.d. *List of References on Constraint Handling Techniques used with Evolutionary Algorithms.* [Online] Available at: http://www.cs.cinvestav.mx/~constraint/ [Accessed 2018].

Deb, K., Mohan, M. & Mishra, S., 2003. *A Fast Multiobjective Evolutionary Algorithm for Finding Well Spread Pareto Optimal Solutions,* Kanpur: KanGAL Report 2003002.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan , T., 2002. A fast end elitist multiobjective genetic algorithm: NSA-II. *IEEE Transaction on Evolutionary Computation ,* 6(2), pp. 181-197.

Dechter, I. & Mairi, R., 1991. Temporal constraint network. *Artificial Intelligence ,* Volume 40, pp. 61-65.

Delchambre, A. & Rekiek, B., 2006. *Assembly Line Design: The Balancing of Mixed Model Hybrid Assembly Lines with Genetic Algorithms.* London: Springer Series in Advanced Manufacturing .

Dhungana, A., Falkner, A., Haselbock, A. & Taupe, R., 2017. *Enabling Integrated Product and Factory Configuration in Smart Production Ecosystems.* Viena , IEEE.

Djefel , M., 2010. *Couplage de la configuration de produit et de project de réalisation: exploitation des approches par contraintes et des algorithmes évolutionnaires. Doctoral Thesis ,* Toulouse : Université de Toulouse. Mines Albi .

Dou, R., Zong, C. & Li, M., 2016. An interactive genetic algorithm with the interval arithmetic based on hesitation and its application to achieve customer collaborative product configuration design. *Applied Soft Computing,* Volume 38, pp. 384-394.

Dubitzky, W., Wolkenhauer, O., Cho, K. & Yokota, H., 2013. *Encyclopedia of Systems Biology.* New York: Springer.

Du, G., Jiao , R. & Chen, M., 2014. Joint optimization of product family configuration and scaling design by stackelberg game. *European Journal of Operational Research ,* 232(2), pp. 330-341.

Eiben, A., 2001. Evolutionary Algorithms and constraint satisfaction: Definition, survey, methodology and research directions. In: *Theoretical Aspects of Evolutionary Computing. Natural Computing Series.* Berlin Heidelberg: Springer, pp. 13-58.

Felfernig, A., Holtz, L., Bagley, C. & Tiihonen , J., 2014. Chapter 6: Configuration Knowledge Representation and Reasoning . In: *Knowledge Based Configuration: From Research to Business Cases* . s.l.:Elsevier/Morgan Kaufmann, pp. 41-72.

Felfernig, A., Hotz, L., Bagley, C. & Tiihonen, J., 2014. Chapter 1: Motivation for the book. In: *Knowledge Based Configuration: From Research to Business Cases.* Waltham: Elsevier/Morgan Kuafmann, pp. 03-07.

Felfernig, A., Hotz, L., Bagley, C. & Tiihonen, J., 2014. Chapter 11: Knowledge Engineering for Confguration Systems. In: *Knowledge Based Configuration: From Research to Business Cases.* Waltham: Elsevier/Morgan Kaufmann, pp. 139-155.

Felfernig, A., Hotz, L., Bagley, C. & Tiihonen, J., 2014. Chapter 2: A Short History of Configuration Technologies. In: *Knowledge Based Configuration: From Research to Business cases.* Waltham : Elsevier/Morgan Kaufmann, pp. 9-17.

Fogel , L., Owens, A. & Walsh, M., 1966. *Artificial Intelligence Through Simulated Evolution.* s.l.:John Wiley & Sons.

Fonseca, C. & Flemming, P., 1993. *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization.* San Francisco, Morgan Kaufmann Publishers Inc..

Freuder, E., 1997. In pursuit of the Holy Grail. *Constraints ,* 2(1), pp. 57-61.

Gelle, E. & Faltings, B., 2003. Solving Mixed and Conditional Constraint Satisfaction Problems. In: *Constraints.* Dutch: Kluwer Academic Publishers, pp. 107-141.

Gelle, E. & Weigel, R., 1995. Interactive Configuration Based on Incremental Constraint Satisfaction. In: *Knowledge Intensive CAD. KIC 1995. IFIP Advances in Information and Communication Technology.* Boston : Springer , pp. 117-126.

Gero, J., 1990. Design prototypes: a knowledge representation schema for design. *A.I. Magazine ,* 11(4), pp. 26-36.

Ghédira , K., 2013. *Constraint Satisfaction Problems. CSP Formalisms and Techniques.* Great Britain/United States : ISTE Ltd and John Wiley & Sons, Inc.

Gottschalk, F. & La Rosa, M., 2010. Modern Business Process Automation: YAWL and its support enviroment . In: *Modern Business Process Automation .* Berlin, Heidelberg: Springer, pp. 459-487.

Gottschalk, F. et al., 2009. Configurable process models: experiences from a municipality case study. In: *Advanced Information Systems Engineering. 21st International Conference, CAiSE.* Amsterdam, The Netherlands: Springer , pp. 486-500.

Grzechca , W., 2011. *Assembly Line-Theory and Practice.* Croatia: InTech Open Access Publisher .

Gupta , S. & McGovern, S., 2011. Chapter 10: Combinatorial Optimization Searches. In: *Disassembly Line: Balancing and Modeling.* New York: The McGraw-Hill Companies, Inc..

Hamdy, M., Hasan, A. & Siren, K., 2013. A multistage optimization method for cost optimal and nearly zero energy building solutions in line with the EPBD recast 2010. *Energy and Buildings ,* Volume 56, pp. 189-203.

Han, S. & Lee, J., 2011. Knowledge based configuration design of a train bogie. *Journal of Mechanical Science and Technology ,* 24(12), pp. 2503-2510.

Hofstedt, P. & Schneeweiss, D., 2013. *A constraint Based Interactive Product Configurator.* Berling , Springer .

Holland , J., 1992. *Adaptation in Natural and Artificial Systems.* Cambridge, MA: MIT Press.

Hong, G., Xue, D. & Tu, Y., 2010. Rapid identification of the optimal product configuration and its parameters based on customer centric product modeling for one of a kind production. *Computers in Industry ,* 61(3), pp. 270-279.

Hoos, H. & Stützle, T., 2004. *Stochastic Local Search Foundations and Applications,* San Francisco : Morgan Kaufmann.

Huang, H. & Gu, Y., 2006. Development mode based on integration of product models and process models. *Concurrent Engineering: Research and Applications,* 14(1), pp. 27-34.

Jensen , R. & Lars, S., 2005. *Power Supply Restoration, Master Thesis ,* Copenhagen : IT University of Copenhagen .

Jiang, Z., Sisi, X., Lin, L. & Zhaoqian, L., 2011. Inventory shortage driven optimisation for product configuration variation. *International Journal of Production Research ,* 49(4), pp. 1045-1060.

Jiao, J., Simpson , T. & Siddique , Z., 2007. Product family design and platform based product development : a state of the art review. *Journal of Intelligent Manufacturing ,* 18(1), pp. 5-29.

Jiao, J. & Zhang, Y., 2005. Product portfolio planning based on customer engineering interaction. *IIE Trans,* 37(9), pp. 801-814.

Ji, J., Yu, W. & Zhang, J., 2017. *A two stage coevalution approach for constrained optimization.* Berlin , ACM , pp. 167-168.

Kaiser, A., Wolfgang, K. & Carsten, S., 2003. Formal Methods for the Validation of Automotive Product Configuration Data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing ,* 17(2).

Kopisch, M. & Gunter, A., 1992. Configuration of a passenger aircraft cabin based on conceptual hierarchy, constraints and flexible control.. In: *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* Berling, Heidelberg: Springer.

Krokhin, A. & Zivny, S., 2017. *The constraint satisfcation problem: Complexity and Approximability.* Germany: Dagstuhl Publishing.

Li, L., Chen , L., Huang, Z. & Zhong, Y., 2006. Product Configuration Optimization Using a Multiobjective GA. *International Journal of Advanced Manufacturing Technology ,* Volume 30, pp. 20-29.

Lindemann, U., 2007. *A vision to overcome chaotic design for X processes in early phases.* France, Design Society .

Linder , J., Lindkvist, S. & Sjoberg, J., 2012. *Two Step Framework for Interactive Multiobjective Optimization. Technical Report LiTH-ISY-R-3043,* Linkoping: Department of Electrical Engineering. Linkoping University.

Liu, Y., Zhang, Z. & Liu, Z., 2011. Customized configuration for hierarchical products: component clustering and optimization. *International Journal of Advanced Manufacturing Technology ,* 57(9-12), pp. 1223-1233.

López Jaimes, A. & Coello Coello, C., 2013. Interactive Approaches to Multiojective Evolutionary Algorithms. Chap. 8. In: *Mulricriteria Decision Aid and Artificial Intelligence: Links.Theory and Applications.* West Sussex: Wiley Blackwell, pp. 191-208.

Macdonald , K. & Prosser, P., 2002. *A case study of constraint programming for configuration problems. Rapp Tech. APES-45-2002,* s.l.: APES Research Group.

Marti, M., 2007. *Complexity Management. Optimizing Product Architecture of Industrial Products,* s.l.: Deutscher Universitäts-Verlag.

Mayer, W., Stumptner, M., Killisperger, P. & Grossman, G., 2011. A declarative framework for work process configuration. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing ,* 25(2), pp. 143-162.

McDermott, J., 1982. R1: a rule based configurer of computer systems. *Artificial Intelligence ,* 19(1), pp. 39-88.

Meiri, I., 1996. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence ,* 87(1-2), pp. 343-385.

Meyer, M. & Lehnerd, A., 1997. The power of product platforms: Building value and cost leadership. *Free Press.*

Miettinen, K., Ruiz|, F. & Wierzbicki, A., 2008. Introduction to Multiobjective Optimization: Interactive Approaches. *Multiobjective Optimization. Lecture Notes in Computer Science ,* Volume 5252, pp. 27-57.

Mittal, S. & Falkenhainer, B., 1990. *Dynamics constraint satisfaction problems.* Boston, Massachusetts, AAAI Press, pp. 25-32.

Mittal, S. & Frayman, F., 1989. *Towards a generic model of configuration tasks.* Detroit, Morgan Kaufmann Publishers Inc.

Montanari, U., 1974. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences ,* Volume 7, pp. 95-132.

Monz, M., Küfer, K., Bortfeld, T. & Thieke, C., 2008. Pareto Navigation: Algorithm Foundation of Interactive Multicriteria IMRT Planning. *Physics in Medicine and Biology,* Volume 53, pp. 985-998.

Mouhoub, M. & Sukpan, A., 2005. *A new temporal CSP Framework Handling Composite Variables and Activity Constraints.* Hong Kong, IEEE, pp. 143-149.

Mulyanto , T., 2002. *Utilisation des techniques de programmation par contraintes pour la conception d´avions. Doctoral Thesis ,* Toulouse : Écoles Nationale Supérieure de l´Aéronautique et de l´Éspace.

Pine , B. J., 1993. Mass Customization: The New Frontier in Business Competition. *Harvard Business School Press.*

Pitiot, P., Aldanondo , M. & Vareilles, E., 2014. Concurrent product configuration and process planning: Some optimization experimental results. *Computers in Industry ,* 65(4), pp. 610-621.

Pitiot, P. et al., 2013. Concurrent product configuration and process planning, towards an approoach combining interactivity and optimality. *International Journal of Production Research ,* 51(2), pp. 524-541.

Pitiot, P. et al., 2019. Optimization of the concurrent product and process configuration: an approach to reduce computation time with an experimental evaluation. *International Journal of Production Research.*

Rechenberg, I., 1965. *Cybernetic solution path of an experimental problem.* Farnborough: Royal Aircraft Establishment Library Translation.

Robertson , D. & Ulrich, K., 1998. Planning of product platforms. *Sloan Management Review,* 39(4), pp. 19-31.

Sabin , D. & Weigel, R., 1998. Product Configuration Framework- A Survey. *IEEE Intelligent Systems ,* 13(4), pp. 19-31.

Sabin, D. & Freuder, E., 1996. *Configuration as Composite Constraint Satisfaction.* California, AAAI Press, pp. 153-161.

Schierholt, K., 2001. Process configuration: combining the principles of product configuration and process planning. *AI EDAM,* 15(5), pp. 411-424.

Schierholt, K., 2001. *Process Configuration: Mastering Knowledge intensive planning tasks,* Zurich : Hochschulverlag AG, ETH Zurich .

Sinha, A., Korhonen , P., Wallenius , J. & Deb, K., 2014. An Interactive Evolutionary Multiobjective Optimization Algorithm with a Limited Number of Decision Maker Calls. *European Journal of Operational Research ,* 233(3), pp. 674-688.

Sinz, C., Kaiser, A. & Küchlin, W., 2003. Formal methods for the valdation of automotive product configuration data.. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* Volume 17, pp. 75-97.

Soininen , T. & Niemela , T., 1999. *Developing a declarative rule language for application in product configuration.* Berlin , Springer .

Soininen, T., Tiihonen, J., Mannisto, T. & Sulonen, R., 1998. Towards a General Ontology of Configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 12(4), pp. 357-372.

Soloway , E., Bachant, J. & Jensen, K., 1987. Assessing the maintainability of XCON in RIME: coping with problem of very large rule bases.. *Proceedings of the Sixth National Conference on Artifical Intelligence (AAAI-87),* pp. 113-125.

Song, W. & Chan, F., 2015. Multiobjetive configuration optimization for product extension service. *Journal of Manufacturing Systems ,* 37(Part I ), pp. 113-125.

Srinivas, N. & Deb, K., 1994. Multiobjective function optimisation using nondominated sorting in genetic algorithms. *Evolutionary Computation ,* Volume 2, pp. 221-248.

Stewart, D., 1981. The design structure system. A method for managing the design of complex systems.. *IEEE Transactions on Engineering Management ,* Volume 28, pp. 71-74.

Stumptner , M., Friedrich, G. & Haselböck, A., 1998. Generative Contraint Based Configuration of Large Technical Systems. *Artifical Intelligence for Engineering Design, Analysis and Manufacturing,* pp. 307-320.

Stumptner, M. & Haselböck, A., 1993. A generative constraint formalism for configuration problems. In: *Advances in Artificial Intelligence* . Berlin Heidelberg : Springer , pp. 302-313.

Subbarayan, 2006. *http://www.itu.dk/research/cla/externals/clib/.* [Online].

Suh, N., 1990. The principes od design. *Oxford series and advanced manufacturing. Oxford University Press.*

Suh, N., 2001. Axiomatic Design: Advances and Applications.. *Oxford University Press* .

Tang, D., Wang, Q. & Ullah, I., 2017. Optimisation of product configuration in consideration of customer satisfaction and low carbon. *International Journal of Production Research,* 55(12), pp. 3349-3373.

Tiihonen , J., Mayer, W., Stumptner, M. & Heiskala, M., 2014. Chapter 21: Configuring Services and Processes. In: *Knowledge Based Configuration: From Research to Business Cases.* Waltham: Elsevier/Morgan Kaufmann, pp. 251-260.

Tsamardinos , I., Vidal , T. & Pollack, M., 2003. CTP: A new Constraint Based Formalism for Conditional Temproal Planning. In: *Constraints 8.4.* USA: Springer, pp. 365-388.

Tsang, E., 1993. *Foundations of Constraints Satisfaction.* UK/USA: Academic Press Limited .

Ulrich , K. T. & Eppinger, S., 1995. *Product Design and Development.* USA: McGraw Hill.

Ulrich, K., 1995. The role of product architecture in the manufacturing firm. *Research Policy 24. Elsevier Science B.V..*

Van Oudenhove de Saint Gery , T., 2006. *Contribution à l´élaboration d´un formalisme gérant la pertinence pour les problèmes d´aide à la conception à base de constraintes. Doctoral Thesis. ,* Toulouse : Institut National Polytecnique de Toulouse .

Vareilles , E., 2005. *Conception et approches par propgation de contraintes: contribution à la mise en ouevre d´un outils d´aide intéractif. Doctoral Thesis,* Toulouse: Institut Nationale Polytechnique de Toulouse.

Vareilles, E., 2015. *Configuration interactive et contraintes: Connaissances, Filtrage et Extensions. Mémoire d´HDR,* Albi France: Écoles de Mines .

Vareilles, E. et al., 2012. *CoFiADe Constraint Filtering for Aiding Design.* Toulouse , JFPC 2012.

Veron, M., 2001. *Modélisation et résolution du problème de configuration industrielle: utilisation des techniques de satisfaction de contraintes. Doctoral Thesis ,* Toulouse : Institut Polytechnique de Toulouse .

Veron, M., Fargier, H. & Aldanondo, M., 1999. *From CSP to Configuration Problems.* California , AAAI.

Vilim, P., Bartak , R. & Cepek, O., 2004. *Unary ressource constraint with optional activities.* Berlin, Springer, pp. 62-76.

Viswanathan , S. & Allada, V., 2006. Product configuration optimization for disassembly planning: A differential approach. *In Omega ,* 34(6), pp. 599-616.

Wagner, T., Beume, N. & Naujoks , B., 2007. Pareto, Aggregation and Indicator Based Methods in Many Objective Optimization. In: *Evolutionary Multicriterion Optimization. EMO 2007. Lecture Notes in Computer Science* . Berlin : Springer .

Wang, L., Sheng Zhong, S. & Jian Zhang , Y., 2015. Process configuration based on generative constraint satisfaction problem. *Journal of Intelligence Manufacturing ,* 28(4), pp. 945-952.

Wang, Y. & Tseng, M., 2012. A Two Step Hierarchical Product Configurator Design Methodology . In: *Foundations of Intelligent Systems. ISMIS 1012. Lecture Notes in Computer Science* . Berlin, Heidelberg: Springer .

Wei, W., Fan, W. & Li, Z., 2014. Multiobjective optimization and evaluation method of modular product configuration design. *International Journal of Advanced Manufacturing Technology ,* 75(9-12), pp. 1527-1536.

Wu, D., Zhang, L. & Jiao, R., 2013. SysML based design chain information modeling for variety management in production reconfiguration. *Journal of Intelligent Manufacturing ,* 24(3), pp. 575-596.

Xu, Z., 2005. Concurrent optimization of product module selection and assembly line configuration: A multiobjective approach. *Journal of Manufacturing Science and Engineering,* 127(4), pp. 875-884.

Yadav, A., Kusum, D. & Sushil, K., 2012. An Harmonic Potential Well Based Particle Swarm Optimization. *Journal of Information and Operations Management,* Volume 3.

Yannou, B., 1998. Les apports de la programmation par contraintes en conception. In: *Conception de produits mécaniques: méthodes, modèles et outils.* Paris: D´HERMÉS, pp. 457-486.

Zhang , L., 2007. *Process platform based production configuration for mass customization. Doctoral dissertation ,* Singapore : University of Singapore.

Zhang , L., Vareilles, E. & Aldanondo , M., 2013. Generic bill of functions, materials and operations for SAP2 configuration.. *International Journal for Production Research ,* 51(2), pp. 465-478.

Zhang , Q. & Li, H., 2007. MOEA/D: A Multi-Objective Evolutionary Algorithm Based on Decomposition. *IEEE Translation on Evolutionary Computation ,* 11(6), pp. 712-731.

Zhang, L., 2014. Product Configuration: A review of the state of the art and future research. *International Journal for Production Research ,* 52(21), pp. 6381-6388.

Zhang, L. & Rodrigues, B., 2010. Nested couloured timed Petri nets for production configuration of product families. *International Journal for Production Research ,* 48(6), pp. 1805-1833.

Zhang, L., Xu, Q., Yu, Y. & Jiao, R., 2012. Domain based production configuration with constraint satisfaction. *International Journal for Production Research,* 50(24), pp. 7149-7166.

Zhou, C., Lin, Z. & Liu, C., 2008. Customer driven product configuration optimization for assemble to order manufacturing enterprises. *International Journal of Advanced Manufacturing Technology ,* 38(1-2), pp. 185-194.

Zitzler, E., Laumanns, M. & Thiele , L., 2002. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization.* Zurich, ETH, pp. 95-100.

Zitzler, E. & Thiele, L., 1998. *Multiojective optmization using evolutionary algorithms-a comparative case study.* Berlin, Springer , pp. 291-301.

# Appendices

## Appendix 1: Specification of 8 Problem Cases

## CASE 1: PLATFORM_MEDIUM_MEDIUM

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 1: Platform_Medium_Medium** which is a platform architecture model is showed on figure 42. This reference case gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 24 configuration variables in product side (14 fdv and 10 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 9 PCEP patterns (three patterns for each module),
- 26 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 26 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

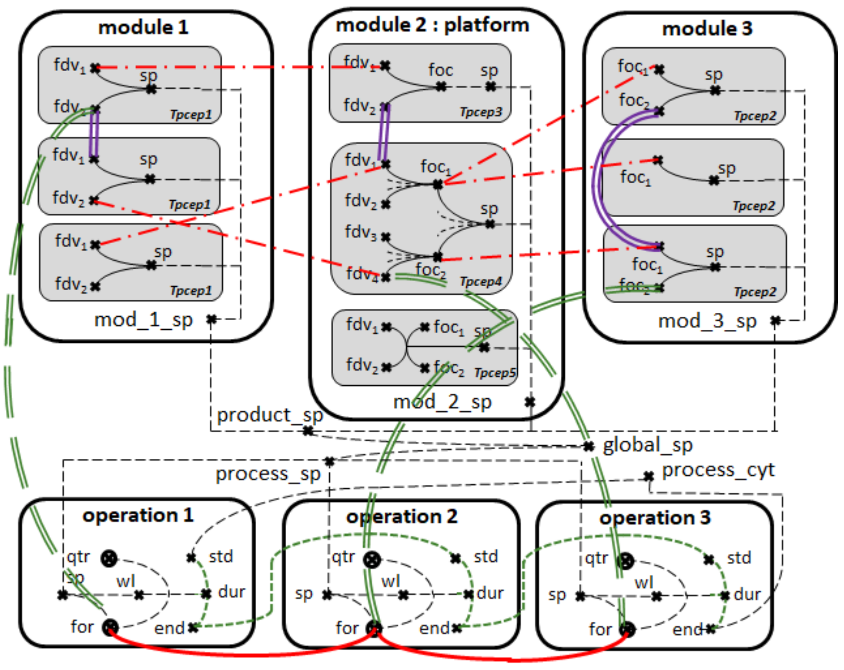


*Figure 42-Case 1 Platform_Medium_Medium*

One module, module 1, has a functional description with only Tpcep1 pattern (only functional description and selling price variables). Another module, module 3, has component description with only Tpcep2 patterns (only family of components and selling price variables). While the platform module, module 2, gathers a selection of physical-functional description with mixed patterns. Each module is linked to one operation by a coupling configuration constraint and each operation is linked to another operation by a configuration constraint.

## CASE 2: MODULAR_MEDIUM_MEDIUM

### A. MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 2: Modular_Medium_Medium** which is a modular architecture model is showed on figure 43. This reference case gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 24 configuration variables in product side (18 fdv and 6 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 9 PCEP patterns (three patterns for each module),
- 26 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 26 evaluation constraints (constraints needed to compute selling price and cycle time).

### B. DIAGRAM OF THE MODEL

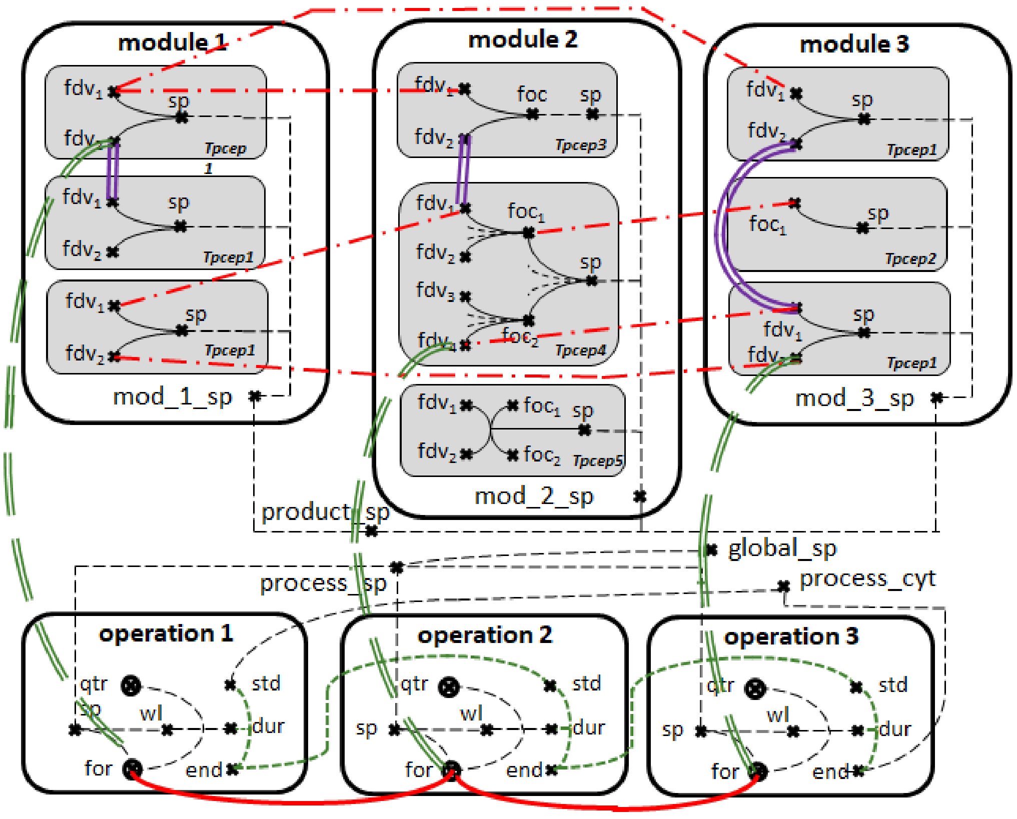


*Figure 43-Case 2:Modular_Medium_medium*

One module, module 1, has a functional description with only Tpcep1 pattern (only functional description and selling price variables). The module 3, has a physical-functional description with two Tpcep1 patterns and one Tpcep2 pattern. Module 2 gathers a selection of physical-functional description with mixed patterns (Tpcep3, Tpcep4 and Tpcep5). There is an interaction between all the modules. Each module is linked to one operation by a coupling configuration constraints and each operation is linked to another operation by a configuration constraint.

## CASE 3: INTEGRATED_MEDIUM_MEDIUM

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 3: Integrated_Medium_Medium** which is a special modular architecture model is showed on figure 44. This reference case gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 24 configuration variables in product side (18 fdv and 6 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 9 PCEP patterns (three patterns for each module),
- 26 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 26 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

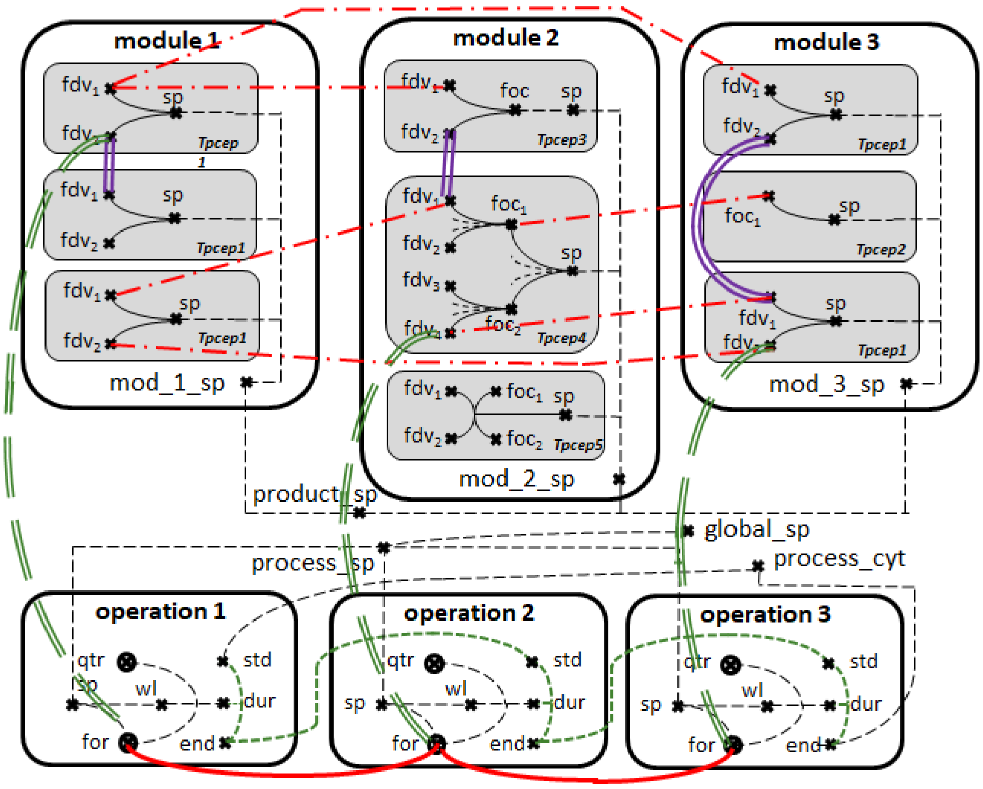


*Figure 44-Case 3:Integrated_Medium_Medium*

One module, module 1, has a functional description with only Tpcep1 pattern (only functional description and selling price variables). The module 3, has a physical-functional description with two Tpcep1 patterns and one Tpcep2 pattern. Module 2 gathers a selection of physical-functional description with mixed patterns (Tpcep3, Tpcep4 and Tpcep5). There is an interaction between all the modules. The relationships between modules (inter-module constraint) have high density. Each module is linked to one operation by a coupling configuration constraints and each operation is linked to another operation by a configuration constraint.

## CASE 4: PLATFORM_SMALL_MEDIUM

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 4: Platform_Small_Medium** which is a platform architecture model is showed on figure 45. This reference case gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 9 configuration variables in product side (5 fdv and 4 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 4 PCEP patterns (one in module 1, two in module 2, one in module 3),
- 12 configuration constraints (3 intra-PCEP, 2 intra-module, 2 inter-module, 3 coupling product/process and 2 inter-operation)
- 21 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

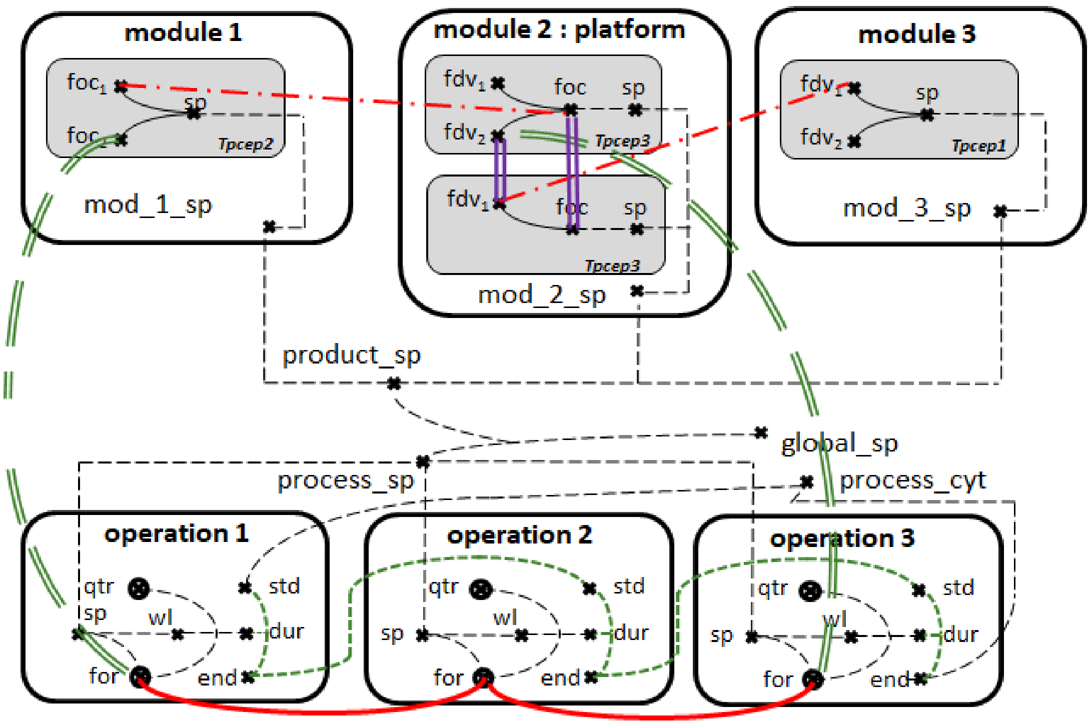


*Figure 45-Case 4: Platform_Small_Medium*

One module, module 1, has one component description with a Tpcep2 pattern (only one family of component and selling price variable). Another module, module 3, has one functional description with a Tpcep1 pattern (only one functional description and selling price variable). While the platform module, module 2, gathers physical-functional description with two Tpcep3 patterns. Module1 and module 2 are linked to one operation by a coupling configuration constraint and each operation is linked to another operation by a configuration constraint.

## CASE 5: PLATFORM_INTERMEDIATE_MEDIUM

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 5: Platform_Intermediate_Medium** which is a platform architecture model is showed on figure 46. This reference case gathers:

- 7 modules
- 7 operations (in a serial architecture),
- 46 configuration variables in product side (30 fdv and 16 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 14 configuration variables in process side (7 for and 7 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 18 PCEP patterns (twoo patterns for each module and six for the platform),
- 51 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 51 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

Four modules (Module 1, 3, 5 and 6), have a functional description with only Tpcep1 pattern (only functional description and selling price variables. Three modules (module 2 and 4), have component description with only Tpcep2 patterns (only family of components and selling price variables). While the platform module, module 7, gathers a selection of physical-functional description with mixed patterns (Tpcep3, Tpcep4 and Tpcep5). Six modules (Module 1, 3, 4, 5, 6 and the Platform) are linked to one operation by a coupling configuration constraint and each operation is linked to another operation by a configuration constraint.

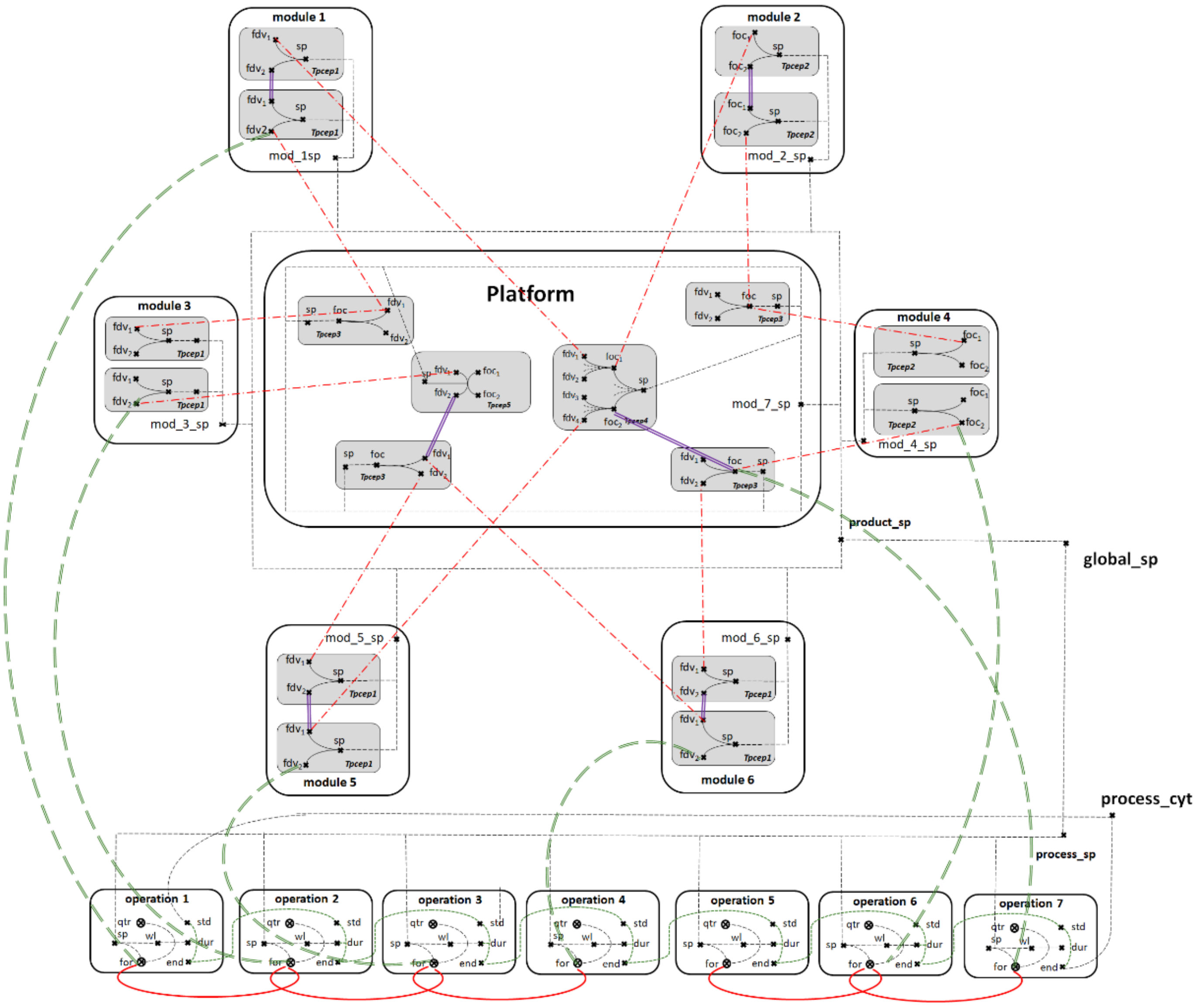


*Figure 46-Case 5: PLATFORM_INTERMEDIATE_MEDIUM*

## CASE 6: PLATFORM_LARGE_MEDIUM

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 6: Platform_Large_Medium** which is a platform architecture model is showed on figure 47. This reference case gathers:

- 10 modules
- 10 operations (in a serial architecture),
- 79 configuration variables in product side (46 fdv and 33 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 20 configuration variables in process side (10 for and 10 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 29 PCEP patterns (twoo patterns for each module and eleven for the platform),
- 82 configuration constraints (33 intra-PCEP, 10 intra-module, 20 inter-module, 10 coupling product/process and 9 inter-operation)
- 74 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

Five modules (Module 1, 3, 5, 7 and 9), have a functional description with only Tpcep1 pattern (only functional description and selling price variables. Four modules (module 2, 4, 6 and 8), have component description with only Tpcep2 patterns (only family of components and selling price variables). While the platform module, module 10, gathers a selection of physical-functional description with mixed patterns (Tpcep3, Tpcep4 and Tpcep5). The ten modules are linked to one operation by a coupling configuration constraint and each operation is linked to another operation by a configuration constraint.

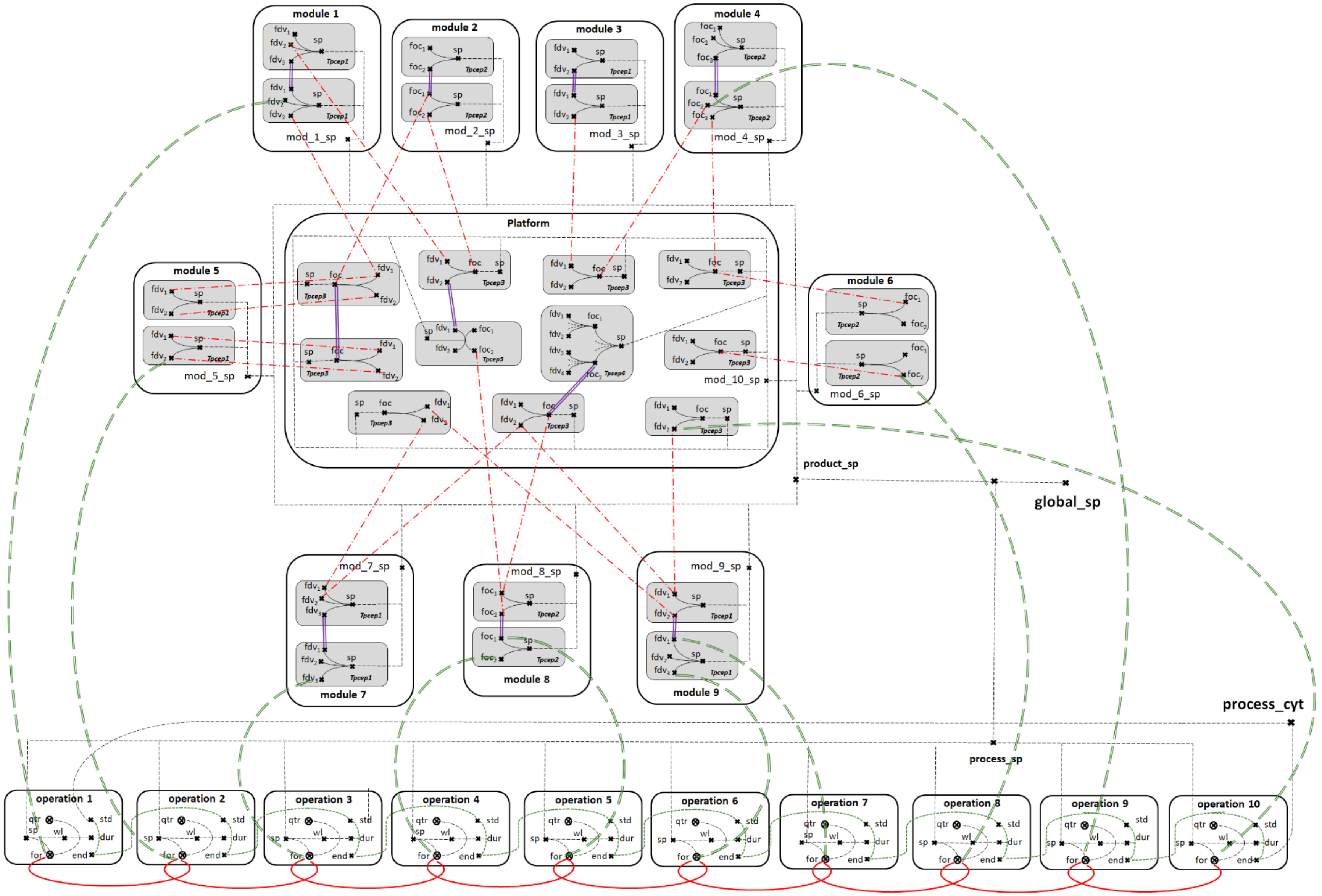


*Figure 47- CASE 6: PLATFORM_LARGE_MEDIUM*

## CASE 7: PLATFORM_MEDIUM_HIGH

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 7: Platform_Medium_High** which is a platform architecture model is showed on figure 48. This reference case gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 24 configuration variables in product side (14 fdv and 10 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 9 PCEP patterns (three patterns for each module),
- 26 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 26 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

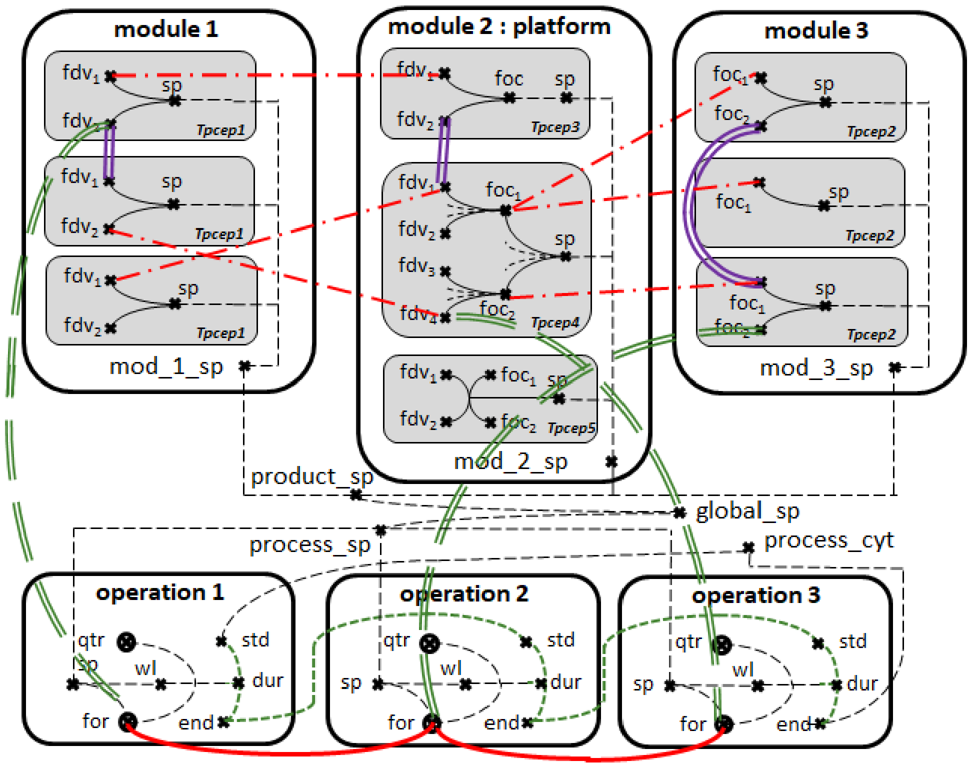


*Figure 48- Case 7:Platform_Medium_High*

One module, module 1, has a functional description with only Tpcep1 pattern (only functional description and selling price variables). Another module, module 3, has component description with only Tpcep2 patterns (only family of components and selling price variables). While the platform module, module 2, gathers a selection of physical-functional description with mixed patterns. Each module is linked to one operation by a coupling configuration constraints and each operation is linked to another operation by a configuration constraint. **This model is similar to the first one (Case 1: Platform_Medium_Medium except by the constraint densities. In this case all the configuration constraints have high density.**

## CASE 8: PLATFORM_MEDIUM_LOW

### A) MAIN CHARACTERISTICS OF THE REFERENCE CASE

The main characteristics of the **Case 8: Platform_Medium_Low** which is a platform architecture model is showed on figure 49. This reference case gathers:

- 3 modules
- 3 operations (in a serial architecture),
- 24 configuration variables in product side (14 fdv and 10 foc), each variable has 6 values in its definition domain (solution space size without constraint around $10^{18}$).
- 6 configuration variables in process side (3 for and 3 qtr), each variable has also 6 values in its definition domain (solution space for whole model around $10^{23}$).
- 9 PCEP patterns (three patterns for each module),
- 26 configuration constraints (12 intra-PCEP, 3 intra-module, 6 inter-module, 3 coupling product/process and 2 inter-operation)
- 26 evaluation constraints (constraints needed to compute selling price and cycle time).

### B) DIAGRAM OF THE MODEL

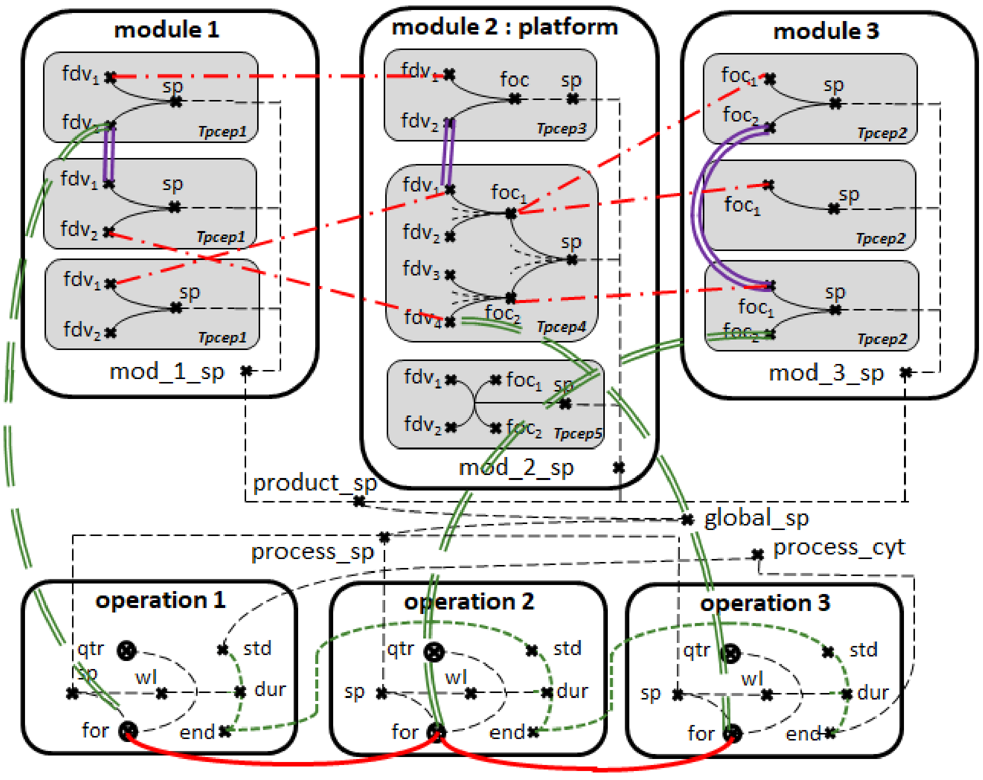


*Figure 49- Case 8:Platform_Medium_Low*

One module, module 1, has a functional description with only Tpcep1 pattern (only functional description and selling price variables). Another module, module 3, has component description with only Tpcep2 patterns (only family of components and selling price variables). While the platform module, module 2, gathers a selection of physical-functional description with mixed patterns. Each module is linked to one operation by a coupling configuration constraints and each operation is linked to another operation by a configuration constraint. **This model is similar to the first one (Case 1: Platform_Medium_Medium except by the constraint densities. In this case the internal relationships (intra-PCEP constraints and intra-module constraints) have high density. The inter-modules constraints, coupling constraints and inter-operation constraints have low density.**

**Résumé**

**Configuration à base de connaissances : une contribution à la modélisation générique, à l'évaluation et à l'optimisation évolutionnaire**

Dans un contexte de personnalisation de masse, la configuration concourante du produit et de son processus d'obtention constituent un défi industriel important : de nombreuses options ou alternatives, de nombreux liens ou contraintes et un besoin d'optimisation des choix réalisés doivent être pris en compte. Ce problème est intitulé O-CPPC (Optimization of Concurrent Product and Process Configuration). Nous considérons ce problème comme un CSP (Constraints Satisfaction Problem) et l'optimisons avec des algorithmes évolutionnaires. Un état de l'art fait apparaître : i) que la plupart des travaux de recherche sont illustrées sur des exemples spécifiques à un cas industriel ou académique et peu représentatifs de la diversité existante ; ii) un besoin d'amélioration des performances d'optimisation afin de gagner en interactivité et faire face à des problèmes de taille plus conséquente. En réponse au premier point, ces travaux de thèse proposent les briques d'un modèle générique du problème O-CPPC. Ces briques permettent d'architecturer le produit et son processus d'obtention. Ce modèle générique est utilisé pour générer un benchmark réaliste pour évaluer les algorithmes d'optimisation. Ce benchmark est ensuite utilisé pour analyser la performance de l'approche évolutionnaire CFB-EA. L'une des forces de cette approche est de proposer rapidement un front de Pareto proche de l'optimum. Pour répondre au second point, une amélioration de cette méthode est proposée puis évaluée. L'idée est, à partir d'un premier front de Pareto approximatif déterminé très rapidement, de demander à l'utilisateur de choisir une zone d'intérêt et de restreindre la recherche de solutions uniquement sur cette zone. Cette amélioration entraine des gains de temps de calcul importants.

**Mots Clés :** Évaluation, Modèle générique, Configuration de produit, Configuration de processus, Configuration concurrente, Optimisation évolutionnaire

**Abstract**

**Knowledge-Based Configuration: A contribution to generic modeling, evaluation, and evolutionary optimization**

In a context of mass customization, the concurrent configuration of the product and its production process constitute an important industrial challenge: Numerous options or alternatives, numerous links or constraints and a need to optimize the choices made. This problem is called O-CPPC (Optimization of Concurrent Product and Process Configuration). We consider this problem as a CSP (Constraints Satisfaction Problem) and optimize it with evolutionary algorithms. A state of the art shows that: i) most studies are illustrated with examples specific to an industrial or academic case and not representative of the existing diversity; ii) a need to improve optimization performance in order to gain interactivity and face larger problems. In response to the first point, this thesis proposes a generic model of the O-CPPC problem. This generic model is used to generate a realistic benchmark for evaluating optimization algorithms. This benchmark is then used to analyze the performance of the CFB-EA evolutionary approach. One of the strengths of this approach is to quickly propose a Pareto front near the optimum. To answer the second point, an improvement of this method is proposed and evaluated. The idea is, from a first approximate Pareto front, to ask the user to choose an area of interest and to restrict the search for solutions only on this area. This improvement results in significant computing time savings.

**Keywords:** Evaluation, Generic model, Product configuration, Process configuration, Concurrent configuration, Evolutionary optimization