ÉCOLE CENTRALE DES ARTS
ET MANUFACTURES
« ÉCOLE CENTRALE PARIS »

THÈSE

présentée par

Baris CANBAZ

pour l'obtention du

GRADE DE DOCTEUR

Spécialité : Génie Industriel

Laboratoire d'accueil : Laboratoire Génie Industriel

SUJET :

Preventing and resolving design conflicts for a collaborative convergence in distributed set-based design

Empêcher et résoudre les conflits de conception pour une convergence collaborative en conception distribuée basée sur les ensembles

soutenue le : 16 Septembre 2013

devant un jury composé de :

Jean-François BOUJUT, Professor, Grenoble INP	Examiner
Laurent GENESTE, Professor, Ecole Nationale d'Ingénieurs de Tarbes	Reviewer
Kemper E. LEWIS, Professor, University at Buffalo-SUNY	President
Panos Y. PAPALAMBROS, Professor, University of Michigan	Examiner
Jean-François PETIOT, Professor, Ecole Centrale de Nantes	Reviewer
Bernard YANNOU, Professor, Ecole Centrale Paris	Supervisor
Pierre-Alain YVARS, Associate Professor, Supméca	Co-supervisor

2013ECAP0050

Table of Contents

List of Abbreviations

BDI :	Belief-Desire-Intention
CAD :	Computer-aided Design
CoCSP :	Cooperative Constraint Satisfaction Problem
CP :	Constraint Programming
CSP :	Constraint Satisfaction Problem
DisCSP :	Distributed Constraint Satisfaction Problem
DynCSP :	Dynamic Constraint Satisfaction Problem
GT :	Game Theory
MAS :	Multi-agent System
MDO :	Multidisciplinary Optimization
MOO :	Multi-objective Optimization
NPD :	New Product Development
SBD :	Set-based Design

List of Figures

List of Tables

Foreword

The present dissertation comprises five chapters. In Chapter 1, an introduction of the general context of this research along with its motivations is presented. Chapters 2, 3 and 4 are formalized as three consecutive scientific papers which have been submitted to international journals. These papers are described below:

- Chapter 2: Canbaz B., Yannou B., Yvars P.-A. (2013) "Improving design process performance of distributed design systems with controlling wellbeing indicators of design actors", **submitted to** Journal of Mechanical Design.
- Chapter 3: Canbaz B., Yannou B., Yvars P.-A. (2013) "Preventing design conflicts in distributed design systems composed of heterogeneous design agents", **submitted to** Engineering Applications of Artificial Intelligence.
- Chapter 4: Canbaz B., Yannou B., Yvars P.-A. (2013) "Resolving design conflicts and promoting solidarity in distributed design", **submitted to** IEEE Transactions on Systems, Man, and Cybernetics: Systems.

Finally, Chapter 5 demonstrates the experimentation which was conducted to verify and validate the model presented in the earlier sections.

Given the form with which the dissertation is formalized, two points are to be noted. The first is that Chapters 2, 3 and 4 have their individual references, while the references of Chapters 1 and 5 are listed at the end of the dissertation. Secondly, some repetitions between Chapters 2, 3 and 4 can be witnessed, which is inherent to the dissertation by papers.

Acknowledgements

I would like to thank…

… Bernard Yannou for supervising my Ph.D. study. Without his insight, guidance and support my research would not have converged to this result.

… Pierre-Alain Yvars for his supervision and huge support in CSP. Without his guidance, my C++ codes would have never worked.

… Jean-François Petiot and Laurent Geneste for accepting to review this dissertation.

… the other members of the jury : Jean-François Boujut, Panos Papalambros and especially Kemper Lewis for also coming from so far. I am honored by their interest in my research.

… Audrey Abi Akle for her advices about the Serious Game transformation.

… all the people who participated to the Serious Game experiments.

… Toufic Zaraket for being a friend, helping me with anything I need at any time.

… all my friends and all the people in LGI for their friendship and support.

… specially my father Sabri and my mother Raziye for standing close behind me despite the distance.

Résumé Etendu (extended summary in French)

Les processus de conception des produits et des services complexes nécessitent la collaboration de plusieurs experts de différentes disciplines. Les activités collaboratives et concourantes fournissent l'expertise nécessaire et réduisent les délais de commercialisation. La faisabilité des projets de développement de nouveaux produits est donc assurée, et également leurs probabilités de succès sont augmentées. Puisqu'il peut y avoir une séparation physique entre les experts de la conception et/ou des limites disciplinaires au sein du problème de conception, une approche de conception distribuée peut être adoptée (Sobieszczanski-Sobieski et al., 1984). Dans la conception distribuée, tandis que le problème global de conception est décomposé en sous-problèmes, la responsabilité est décentralisée et distribuée aux sous-systèmes composés d'un ou plusieurs experts (Papalambros et al., 1997). Les sous-systèmes ont un contrôle limité sur les variables de conception et de performance en raison de leurs expertises et responsabilités limitées. Les variables de conception sont pour dimensionner un produit (par exemple la longueur) et les variables de performance sont pour évaluer les performances d'un produit (par exemple la masse). L'objectif ultime de la conception collaborative distribuée est de résoudre les sous-problèmes simultanément de sorte que le problème global converge vers un optimum global (Zheng et al., 2011). Cependant, comme le soulignent Lewis et Mistree (1998), en réalité, il est très peu probable d'obtenir « la concourance véritable », parce que les sous-systèmes ne sont pas indépendants, mais sont liés les uns aux autres par les couplages entre leurs sous-problèmes.

Les incohérences dans le système de conception peuvent provoquer des conflits de conception par les couplages. Les conflits de conception émergent au cours du processus de conception lorsque les acteurs de conception ne sont pas en mesure de satisfaire leurs propres objectifs. Les incohérences peuvent se poser au niveau du problème et du processus. Les incohérences au niveau du problème sont constituées par les objectifs contradictoires des sous-systèmes. Satisfaire l'objectif d'un acteur peut se faire au détriment des autres. Les incohérences au niveau du processus sont constituées par le manque de coordination entre les procédures de travail des sous-systèmes (Zhao and Jin, 2003). Par exemple, un acteur qui modifie le modèle de conception plus fréquemment et de façon restrictive peut bloquer les autres acteurs en essayant de satisfaire leurs propres objectifs de conception. Les conflits de conception s’aggravent lorsque les niveaux de satisfaction des acteurs obtenus à partir de la solution globale divergent. Un acteur peut alors être très satisfait alors que les autres ne le sont pas autant ou sont même insatisfaits.

La divergence des satisfactions représente l'intensité des conflits de conception. L’hypothèse sous-jacente est que les conflits de conception doivent être éliminés systématiquement afin d'améliorer à la fois la performance des processus de conception et la qualité de la conception finale. Détecter et éliminer les conflits à tout moment revient à gérer un compromis collaboratif. L'idéal est d'éviter les conflits le plus en amont possible lors du processus de conception. Si certains conflits de conception n'ont pas pu être évités, ils doivent être détectés et résolus avant qu'ils ne s’amplifient trop. La résolution des conflits intenses peut être très coûteuse en temps car elle met en jeu des boucles de reconception. Mais il est également difficile de détecter et de justifier l’existence de conflits avant qu'ils ne deviennent intenses. L’évitement, la justification et la résolution des conflits de conception sont des concepts indispensables pour obtenir des solutions globalement satisfaisantes pour lesquelles les niveaux de satisfaction des sous-systèmes sont équilibrés.

La technique choisie pour la modélisation du processus de conception affecte la solution collaborative émergeante de différents sous-problèmes. Il y a deux approches principales qui sont adoptées pour définir le processus : descendante et ascendante (Fathianathan and Panchal, 2009). Dans l’approche descendante, les décisions sont prises pour paramétrer les variables de conception afin de trouver des solutions détaillées qui répondent aux objectifs des acteurs de conception. Cette approche est considérée comme une transition d’un niveau abstrait à un niveau détaillé. En revanche, l'approche de conception ascendante consiste à définir des solutions détaillées pour identifier les valeurs des variables de conception. Avec cette approche, les acteurs peuvent prendre des décisions sur leurs variables de performances. L’approche descendante nécessite une décomposition détaillée du problème où toutes les relations entre les variables sont explicites. Toutefois, cela peut ne pas être possible lorsque la complexité du problème de conception est très élevée et le problème contient trop de couplages. Par conséquent, l'effet des décisions concernant les variables de conception sur les variables de performance est très incertain, surtout dans la conception préliminaire.

Un problème général de conception, indépendamment de l'approche adoptée pour le processus, est la présence de l'incertitude épistémique. Cette incertitude est l'imprécision dans le modèle de conception, causée par le manque de connaissance sur la décision finale (Parry, 1996). Particulièrement dans la phase préliminaire de la conception, des valeurs précises ne peuvent pas être attribuées aux variables (Antonsson and Otto, 1995; Yannou, 2004). Les approches déterministes nécessitent d'attribuer des valeurs précises, dites aussi ponctuelles, aux variables afin d'optimiser le modèle. Dans la pratique, cela nécessite de faire des choix de conception inutilement précis sans connaitre les conséquences sur les performances. Par conséquent, les aspects d'incertitude sont mal gérés au détriment de la performance globale. Selon Malak et al. (2009), l'incertitude nécessite la représentation des variables par des intervalles ou des ensembles imprécis.

Un problème de conception à grande échelle contient généralement plusieurs objectifs de conception. L'optimisation multiobjectif est l'approche la plus simple pour évaluer les multiples objectifs. Dans cette approche, on estime qu'il existe toujours une coopération entière entre les acteurs de conception (Jagannatha Rao et al., 1997). Les acteurs sont considérés comme des membres d'une équipe, et ils interagissent de manière explicite afin d'améliorer la solution globale. Une coopération entière est difficile à obtenir dans les systèmes complexes de conception en raison de potentiels obstacles d'information, d'organisation et de processus (Lewis and Mistree, 1998). L'optimisation multidisciplinaire est proposée afin d'intégrer diverses disciplines par des cadres qui comprennent des définitions formelles de communication et de coordination entre les sous-systèmes (Simpson et Martins, 2011). Selon Devendorf et Lewis (2011), cela peut entraîner des coûts élevés de communication, de coordination, de gestion et d'organisation dans les systèmes complexes de conception. Les approches basées sur la théorie des jeux sont proposées pour traiter les problèmes de conception distribuée non-coopérative où la circulation de l'information est imparfaite (Vincent, 1983). Dans ces approches, chaque joueur construit un ensemble de réactions rationnelles (ERR) à l'égard de toute stratégie inconnue faite par un autre joueur. L'intersection des ensembles de réactions rationnelles représente la solution de Nash.

Bien que Lewis et Mistree (1998) mettent en évidence l'extrême difficulté de construire les ERRs exactes pour les problèmes complexes, ils développent des techniques efficaces pour déterminer les ERRs approximatifs dans leur recherche suivante (Lewis et Mistree, 2001). Les systèmes multi-agent (SMA) sont utilisés afin de prendre en compte les interactions dynamiques des agents de conception (Klein 1991, Koulinitch and Sheremetov 1998). Les attitudes de conception ne sont pas considérées dans les SMA proposés, alors que les conflits de conception peuvent être explorés en modélisant ces attitudes. Les méthodes interactives floues sont proposées afin de modéliser les interactions imprécises dans les systèmes de conception. Cependant, selon Yannou et Harmel (2004), la représentation floue des variables n'est pas efficace dans la propagation de l'incertitude autant que la représentation par des intervalles/ensembles imprécis.

« La conception basée sur les ensembles » (en anglais : set-based design, SBD) considère le processus de conception comme une évolution continue des décisions concourantes (Sobek et al., 1999; Ward et al., 1994). Les variables sont représentées par des valeurs imprécises, autrement dit domaines de valeur (ensembles pour les variables discrètes, intervalles pour les variables réelles). L'incertitude épistémique peut donc être propagée et évaluée. Les acteurs restreignent progressivement leur espace de solution chacun de leur coté en partant des bornes des domaines admissibles de leurs variables. Comme le montre la Figure 1, l'espace de solutions restantes est défini par la superposition des choix locaux des concepteurs.

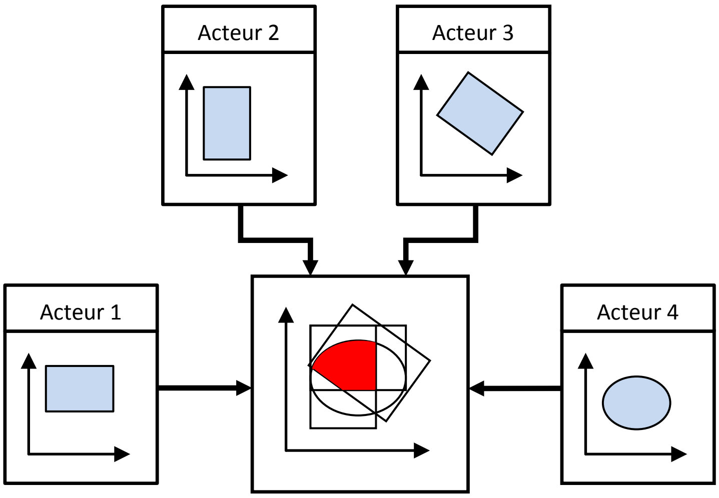


Figure 1: Restriction concourante de l'espace de solution en SBD

Le SBD permet de recueillir l'information avant de prendre des décisions sur le modèle de conception. Les acteurs prennent des décisions au fur et à mesure, en comparant l'espace de solution restant à leurs objectifs. Les décisions sont retardées lorsque l'information n'est pas disponible. Ces décisions qui ont été retardées sont réexaminées à des étapes ultérieures du processus où de plus amples informations ont été recueillies grâce à la réduction de l'incertitude épistémique par des décisions antérieures. L'espace de solution converge donc vers une solution finale progressivement au cours des étapes du processus de conception. Ceci est illustré dans la Figure 2 où les courbes extérieures représentent l'espace de solution, les courbes intérieures représentent les décisions prises à l'étape du processus en considérant l'information émergente de l'étape précédente, et les courbes en pointillés représentent la convergence. Comme démontré par Wang et Terpenny (2003), le SBD offre une souplesse des modifications et une adaptabilité aux changements. La robustesse aux erreurs de conception est aussi assurée comme l'indiquent Parsons et al. (1999). Le temps de traitement est par conséquent réduit en raison d'une diminution des activités répétitives et des bouclages de conception.

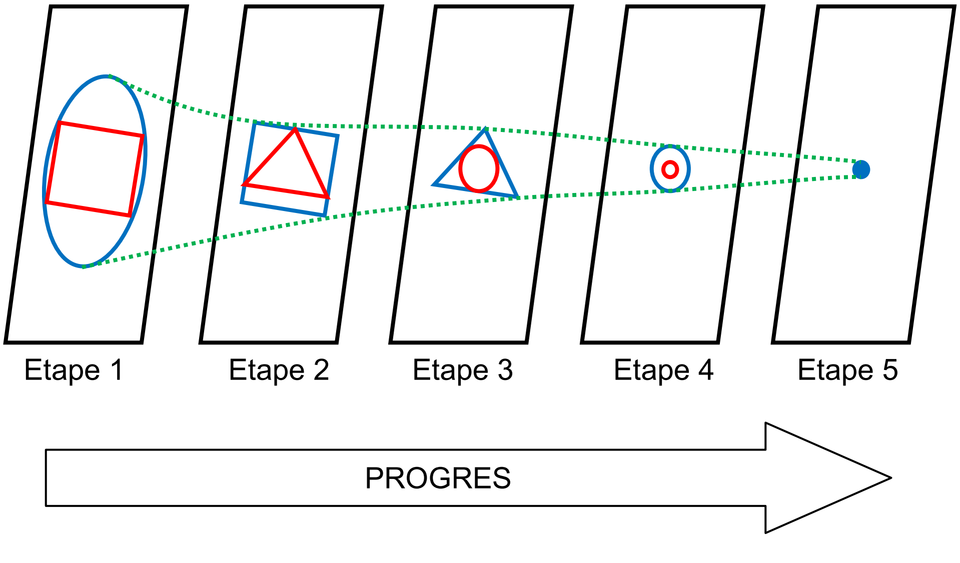


Figure 2: La convergence progressive de l'espace de solution en SBD

Le SBD est adopté avec les techniques de « problème de satisfaction de contraintes » (en anglais : constraint satisfaction problem, CSP) afin de résoudre les problèmes de conception (Meyer and Yvars, 2012; Panchal et al., 2007; Yannou and Harmel, 2006;

Yannou et al., 2013). Un CSP est défini avec trois ensembles : ensemble des variables, ensemble des domaines qui contiennent les valeurs admissibles des variables, et ensemble des contraintes qui limitent le problème (Montanari, 1974). Le produit cartésien des variables définit un espace multidimensionnel qui contient toutes les solutions cohérentes. Ces solutions cohérentes sont les vecteurs des valeurs de variables qui respectent les contraintes. La technique CSP est utilisée comme un outil d'aide à la conception où les décisions de conception sont formalisées par des contraintes qui limitent l'espace de solution au fur à et mesure (Vareilles et al., 2012).

Quand l'incertitude épistémique est réduite, l'espace de solution restant est détecté précisément avec les algorithmes de filtrage de domaine disponibles en programmation par contraintes (en anglais : constraint programming, CP). L'incohérence des contraintes, autrement dit la faisabilité de l'espace de solution, est ainsi vérifiée (Meyer and Yvars, 2012; Yannou and Harmel, 2004). Selon Yannou et Harmel (2004), le CP surpasse les méthodes probabilistes et floues sur la gestion de l'imprécision dans la conception. Le CP permet de définir des contraintes directement sur les variables qui sont des fonctions d'autres variables. Ceci représente un processus ascendant. Par exemple, X et Y sont des variables entières avec des domaines D (X) = [15, 25] et D (Y) = [10, 20] et Z = X x Y est un produit. Si une contrainte est définie sur Z, les valeurs incohérentes de X et Y sont rejetées de leurs domaines. Si Z $\leq$ 200, les domaines des variables sont réduits à D (X) = [15, 20] et D (Y) = [10, 13].

En SBD, la convergence peut être assurée par une boucle séquentielle de fractionnement des intervalles de variables définissant l'espace de solution - l'application de la stratégie round-robin (Granvilliers, 2012) comme le processus de conception -, et par le rejet des contraintes incohérentes (Yvars, 2010, 2009). Cependant, les niveaux de satisfaction de différents acteurs de la conception peuvent encore diverger si la convergence de chaque concepteur n'est pas régulée au sein du groupe. Ce contrôle de convergence collaborative signifie que chaque acteur de conception peut exprimer ses décisions avec la même liberté de conception. Comme le montre la Figure 3, la divergence peut augmenter tout au long du processus de conception. Chaque contrainte rejetée représente un conflit potentiel, et la divergence des niveaux de satisfaction représente l'intensité des conflits.

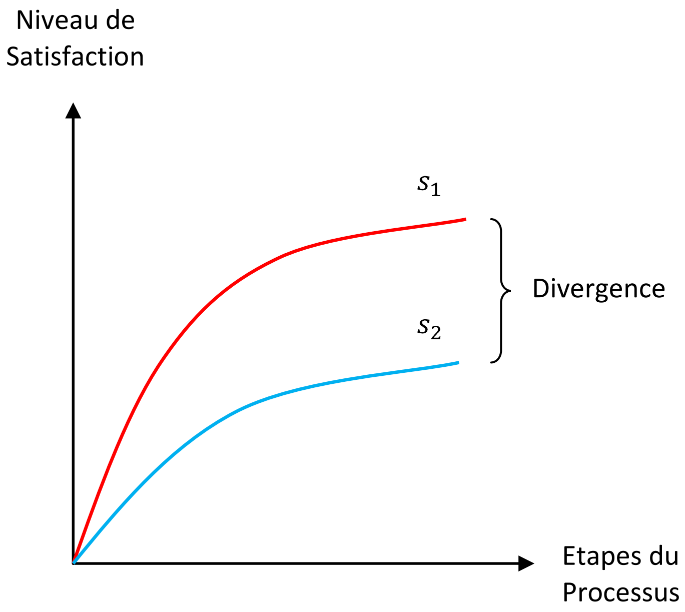


Figure 3: La divergence des niveaux de satisfaction

Les conflits se font au détriment des performances du processus, comme la durée du processus et la satisfaction totale, et aussi de la qualité finale du produit. Ils doivent donc être éliminés systématiquement. Les dynamiques sociales des acteurs de la conception peuvent influencer les conflits (Lu et al., 2000; Pelled et al., 1999). Dans une équipe de conception, les acteurs peuvent refléter des attitudes hétérogènes. La modélisation des attitudes de conception aide à explorer les conflits. Les Systèmes Multi-Agents (SMA) sont une approche connue pour simuler le comportement émergent du système de la conception distribuée (par exemple: Koulinitch and Sheremetov, 1998; Kwon and Lee, 2002). Le système de résolution des conflits peut être adopté avec différentes stratégies qui prennent en compte l'architecture de solidarité des agents. Les stratégies doivent donc être simulées afin de sélectionner la meilleure architecture.

L'objectif de cette thèse est de proposer un ensemble des modèles afin d’empêcher les conflits par rapport aux approches classiques (descendante et ascendante) et de résoudre les conflits qui ne sont pas été évités, tout en surmontant le problème de l'incertitude grâce à l’approche SBD. Les questions de recherche sont les suivantes :

Question 1 : Comment empêcher les conflits de conception dans le SBD distribué tout en améliorant les performances de processus?

Question 2 : Comment modéliser les attitudes de conception et empêcher les conflits dans une conception distribuée composée d'agents hétérogènes?

Question 3 : Comment justifier et résoudre les conflits de conception en SBD distribué?

Question 4 : En quoi la promotion de la solidarité est utile en conception distribuée ?

Afin de répondre à ces questions, nous proposons un ensemble des modèles que nous évoquons brièvement dans ce qui suit.

Les objectifs de conception sont définis en fonction des préférences de produits du marché. Une variable de performance est donc évaluée par les déclarations des préférences afin de déterminer comment son objectif de conception est satisfait. Dans notre modèle, nous combinons les préférences « hard » et « soft » de la programmation physique définie par Messac (1996). Les satisfactions des objectifs de conception reflètent également les satisfactions des acteurs par le modèle de produit. Avec les déclarations des préférences des acteurs, leurs fonctions de satisfaction, $f_s(\ \ )$, sont donc définies. $s_{ki} = f_s(v_i)$ où s_{ki} est la satisfaction de l'acteur k par la variable de performance i, et v_i est la valeur de la variable de performance i. Les préférences « hard » sont : $s_{ki} = 1$ si l'objectif est entièrement satisfait, et $s_{ki} = 0$ si l'objectif est entièrement insatisfait. Les préférences « soft » sont les transitions entre les états entièrement satisfaits et entièrement insatisfaits: $1 > s_{ki} > 0$.

Les satisfactions sont représentées par des intervalles. L'intervalle de satisfaction de l'acteur k, $s_k = [mins_k^t\ , maxs_k^t]$ est dynamique tout au long du processus où $mins_k^t$ est la borne minimale et $maxs_k^t$ est la borne maximale dans l'étape du processus t. Cet intervalle est dynamique, parce qu'il converge avec la progression du modèle pendant les étapes du processus de conception où des contraintes de décision sont ajoutées au fur et à mesure dans le modèle. Dans le processus de conception d'un produit complexe, les activités de conception sont couplées. Les degrés de liberté des acteurs sont donc limités. La borne minimale de l'intervalle de satisfaction est augmentée par les activités de conception de son acteur. Cependant, la borne maximale de l'intervalle est réduite par les activités d'autres acteurs avec des objectifs contradictoires. La convergence est ainsi bilatérale. Au stade final du processus, l'intervalle de satisfaction converge vers une solution où les bornes sont approximativement égales. La Figure 4 explique la convergence bilatérale avec un exemple. C_m et C_n sont les ensembles de contraintes ajoutés au problème par l'acteur k pour augmenter $mins_k$. C'_m et C'_n sont les ensembles de contraintes ajoutées au problème par les autres acteurs de la conception avec des objectifs contradictoires. Ceux-ci réduisent $maxs_k$.

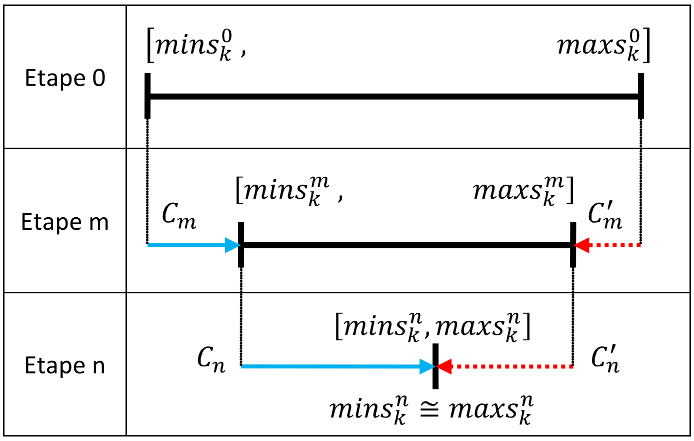


Figure 4: La convergence bilatérale de l'intervalle de satisfaction

Comme la convergence est bilatérale, les acteurs sont obligés de faire des compromis à un certain niveau de satisfaction dans le processus de conception. Un acteur de conception peut définir une préférence pour une valeur de satisfaction pour laquelle il est prêt à accepter un certain compromis. Cette préférence est définie en tenant compte de l'espace de solution, l'incertitude de conception et aussi le degré de liberté des autres acteurs. Elle correspond à la valeur de seuil de compromis T_k. T_k représente la valeur de satisfaction que l'acteur k veut garantir dans l'intervalle s_k. Si $mins_k^t < T_k$, l'acteur de conception définit des contraintes de décision afin d'améliorer s_k. Sinon l'acteur passe à l'état de compromis. Dans ce cas, les acteurs arrêtent alors d'ajouter des contraintes dans le modèle. Cela laisse de la liberté aux autres acteurs, parce que les bornes maximales de leurs intervalles de satisfaction ne sont plus limitées par des acteurs en état de compromis.

T_k est une préférence de processus, différente de préférences de produit. Bien que les préférences de produit définissent les fonctions de satisfaction qui évaluent la faisabilité du produit, les valeurs T_k sont les préférences des acteurs qui évaluent les satisfactions par le processus. Afin de faire une distinction de la satisfaction des objectifs de conception (s_k), nous appelons la satisfaction de processus « bien-être » de l'acteur de conception. s_k est normalisée par T_k, et cela donne l'indicateur de bien-être de l'acteur k $(wb_k = s_k/T_k)$. Cet indicateur est évidemment représenté par un intervalle. Notre modèle consiste à utiliser les intervalles de bien-être afin de contrôler l'espace de solution. Cela représente un processus ascendant étendu.

Nous avons défini une simulation de CSP de notre modèle SBD. L'objectif est de simuler les performances du processus qui est contrôlé par les indicateurs de bien-être, par rapport à certains scénarios qui représentent les pratiques générales de conception descendantes et ascendantes. Quatre cas de simulation représentent ces scénarios. Les Cas 1 et 2 représentent les processus de conception descendants classiques. Le Cas 3 est un processus ascendant conventionnel. Le Cas 4 est notre processus de conception ascendant étendu où l'espace de solution est contrôlé par les indicateurs de bien-être.

- Cas 1: Les joueurs définissent des contraintes de décision sur leurs variables de conception normalisées. Chaque joueur peut définir au maximum une contrainte par itération.

- Cas 2: Les joueurs définissent des contraintes de décision sur toutes leurs variables de conception normalisées. C'est une approche « tout d'un coup » où les joueurs peuvent modifier toutes leurs variables ensembles à chaque itération.

- Cas 3: Les joueurs définissent des contraintes de décision sur leurs variables de performance normalisées.

- Cas 4: Les joueurs définissent des contraintes sur leurs indicateurs de bien-être.

Le processus de simulation est évalué par quatre critères de performance de processus : le nombre d'itérations, le nombre d'échecs, la satisfaction totale des objectifs et la divergence des satisfactions individuelles. Le nombre d'itérations représente la durée du processus. Le nombre d'échecs est le nombre de contraintes rejetées. Cela représente donc le nombre de conflits potentiels. La divergence des satisfactions individuelles représente l'intensité des conflits. Les différences absolues entre les valeurs de chaque paire de satisfactions définissent un vecteur. Dans l'idéal, tous les acteurs devraient satisfaire pleinement leurs objectifs. Leurs niveaux de satisfaction sont donc identiques. Dans l'idéal, pour chaque paire d'acteur, cela représente une différence qui est égale à zéro. La distance euclidienne entre cette solution idéale et le vecteur des différences des satisfactions d'un processus représente l'intensité du conflit émergeant. Les quatre cas de simulation sont comparés par rapport à ces critères.

Nous avons effectué une simulation de Monte Carlo avec le problème de la conception d'un système d'embrayage multidisque dérivé de l'exemple étudié dans (Yannou et al., 2010). Les activités des joueurs et leur séquence sont stochastiques. Les résultats montrent que les performances du processus de conception sont améliorées en contrôlant les indicateurs de bien-être. Le nombre de conflits potentiels et l'intensité des conflits sont réduits, car la domination entre les acteurs est largement évitée. Cela veut dire que les conflits de conception sont empêchés. La satisfaction totale est également améliorée tout en gardant la durée du processus minimale. En conclusion, les acteurs de conception peuvent améliorer leur état de bien-être en équilibre tout en réduisant l'incertitude épistémique par des contraintes de décision cohérentes. Ce modèle représente donc un contrôle plus collaboratif par rapport aux autre cas.

Nous avons étendu notre modèle avec une approche SMA, afin de modéliser les attitudes de conception. Comme le montre la Figure 5, un modèle « Croyances-Désirs-Intentions » (en anglais : Belief-Desire-Intention, BDI (Bratman et al., 1988)) est défini pour explorer les attitudes de conception. Un CSP de conception peut être défini avec trois espaces : l'espace de conception défini par les variables de conception, l'espace de performance défini par les variables de performance de conception, et l'espace de solution qui contient ces deux espaces. L'espace de conception détermine le modèle de conception. Ce modèle est dynamique, car il évolue avec des contraintes de décision ajoutées lors du processus. L'analyse de l'espace de conception stimule les agents. Les bornes maximales des intervalles des variables de performance de conception représentent les meilleurs cas possibles, alors que les bornes minimales représentent les pires cas possibles. Ces cas reflètent les croyances de l'agent de conception sur la façon dont ses variables de performance de conception convergent vers ses objectifs de conception.

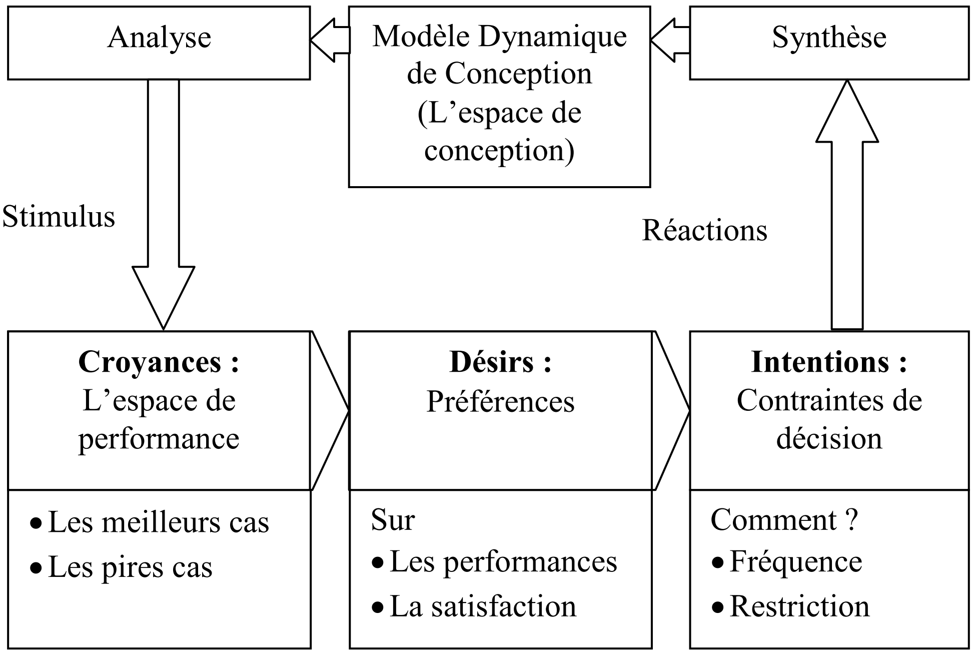


Figure 5: Le modèle BDI des agents de conception

La convergence de la variable de performance d'un agent est bilatérale dans un problème de conception où il y a des objectifs contradictoires. Cette convergence dépend donc des réactions imprévisibles des autres agents, car les agents sont typiquement hétérogènes. L'agent identifie donc certaines préférences pour ses variables de performance et sa satisfaction en tenant compte de ses croyances initiales. Ces préférences reflètent les désirs de l'agent pour la convergence incertaine. Au cours du processus de conception, l'agent peut affirmer ses intentions pour améliorer sa satisfaction en évaluant comment ses croyances instantanées satisfont ses désirs. L'agent réagit aux incertitudes en définissant des contraintes de décision qui limitent l'espace de solution, avec le but d'améliorer les pires cas de ses intervalles. Les intentions sont reflétées par la fréquence et la restriction des définitions des contraintes de décision. La synthèse des réactions crée le modèle modifié.

Un agent k, A_k, est donc défini comme une entité avec quatre attitudes différentes : $A_k\langle Pr_k, T_k, F_k, M_k\rangle$. Pr_k est l'ensemble des préférences de l'agent sur les valeurs de performance. T_k est le seuil de compromis de l'agent. Il représentant la préférence de l'agent sur sa satisfaction. F_k est la fréquence moyenne de l'agent pour définir des contraintes dans le modèle. M_k est le coefficient de restriction des contraintes définies par l'agent, ce qui reflète le caractère restrictif des contraintes. En fonction de leurs attitudes, les agents peuvent avoir des caractères différents. Ils peuvent être plus égoïstes ou plus altruistes par rapport aux autres. Les agents plus égoïstes essaient de satisfaire leurs besoins au maximum sans considérer d'autres agents. Au contraire, les agents plus altruistes considèrent d'autres agents lors qu'ils prennent des décisions.

La Figure 6 représente les caractéristiques égoïstes et altruistes des agents. Lorsque deux agents sont comparés, si les attitudes F_k et M_k sont identiques, l'agent avec le plus grand T_k est plus égoïste que l'autre, car il fait le compromis à une valeur de satisfaction plus grande. Ainsi, il restreint l'espace de solution plus que l'autre, jusqu'à ce que son objectif soit satisfait. Si T_k et M_k sont identiques, l'agent avec le plus grand F_k est plus égoïste que l'autre, parce que quand un agent définit ses contraintes de décision plus fréquemment, il va restreindre l'espace de solution plus rapidement au cours du processus. Par conséquent, il laisse moins d'espace pour les autres agents. Si T_k et F_k sont identiques, l'agent avec le plus grand M_k est plus égoïste que l'autre, parce que les contraintes de décision seront plus restrictives que les contraintes de décision de l'autre agent. Cela permettra de réduire l'espace des solutions au profit de l'agent égoïste.

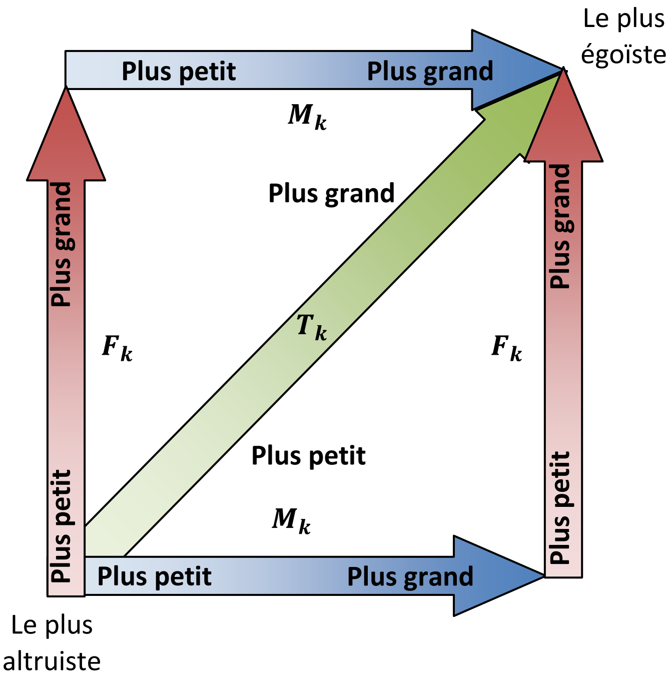


Figure 6: L'égoïsme et l'altruisme des agents de conception

Le caractère d'un agent de conception peut donc être évalué par le modèle BDI. Les interactions des agents avec des caractères hétérogènes peuvent ainsi être simulées. Nous avons étendu le processus ascendant par le modèle BDI. Dans cette approche, les agents affirment des intentions de compromis sur leurs indicateurs de bien-être qui sont dérivés de leurs croyances et désirs. Les simulations CSP sont effectuées avec la méthode Monte Carlo où les caractères des agents sont définis aléatoirement. Le problème de simulation consiste à concevoir un réservoir sous pression (Karandikar and Mistree, 1992; Lewis and Mistree, 1998). Les quatre cas (définis auparavant), représentant les conceptions ascendante et descendante, sont comparés entre eux par rapport aux critères de performance de processus. Les résultats des simulations montrent encore une fois que la performance du processus de conception est significativement améliorée avec cette approche. Dans cette approche, les dominations des agents causées par le processus lui-même sont éliminées. Le nombre de conflits potentiels et l'intensité des conflits sont ainsi réduits. Les résultats des simulations montrent également que l'altruisme réciproque modéré diminue l'intensité des conflits, alors que trop d'altruisme peut diminuer la satisfaction totale obtenue à partir de la solution finale.

Ce modèle a été étendu avec un modèle de gestion des conflits. Il a été développé afin de justifier et puis de résoudre les conflits de conception qui ne peuvent être évités. Si une contrainte est rejetée, il représente un conflit potentiel, puisque les désirs de l'agent qui définit la contrainte ne sont pas satisfaits. S'il y a au moins un autre agent dans un meilleur état de bien-être, le conflit est justifié. C'est parce qu'il est considéré que l'espace de solution n'a pas été restreint en équilibre, en collaboration. C'est-à-dire, au moins un agent a restreint l'espace de solution plus que l'agent en conflit.

Cooperative CSP (CoCSP) est une technique définie par Yvars (2010, 2009) pour obtenir des solutions coopératives dans le SMA. Pour cette technique, si un agent de conception ne peut exercer ses activités de conception, les autres agents peuvent l'aider. Cette aide est effectuée en relaxant certaines décisions qui ont été prises. Un CoCSP est apte à résoudre les problèmes de conception qui sont constitués d'objectifs contradictoires, car il permet aux agents de négocier.

Dans notre modèle, une nouvelle forme de CoCSP a été mise au point afin de résoudre les conflits justifiés. Si la contrainte d'un agent est rejetée, un autre agent peut l'aider en relaxant certaines de ses contraintes qui ont été acceptées. Une nouvelle attitude de conception est donc introduite. Cette attitude, H_k, représente la probabilité d'un agent d'aider les autres. La nouvelle forme d'un agent est la suivante : $A_k\langle Pr_k, T_k, F_k, M_k, H_k\rangle$. Notre modèle de la résolution des conflits détecte l'agent qui peut aider et comment il peut aider de manière optimale. Il est composé de trois phases:

- Phase de négociation : nous proposons qu'un agent puisse aider l'autre en supprimant ses propres contraintes, uniquement si son état de bien-être ne descend pas sous l'état de bien-être de l'agent qui demande de l'aide. En suivant cette procédure, toutes les possibilités d'aide sont détectées.

- Phase de test : les faisabilités des différentes possibilités d'aide sont testées par les techniques de CSP. Parmi toutes les aides possibles, l'aide optimale, qui donne la valeur maximale de la somme de bien-être des agents, est détectée

- Phase d'approbation : l'approbation de l'aide optimale est demandée à l'agent. H_k détermine la probabilité de l'approbation de l'agent *k*.

Les simulations de Monte Carlo de ce modèle sont réalisées avec des caractères d'agents hétérogènes, définis aléatoirement. En outre, certaines expérimentations sont menées avec des agents humains qui s'interagissent dans un jeu sérieux (*serious game*). Les jeux sérieux sont utilisés pour simuler des problèmes complexes avec des joueurs qui n'ont pas l'expertise adéquate au problème (Djaouti et al., 2011). Cela nécessite la transformation du problème complexe en un problème plus divertissant, tout en maintenant son objectif principal (Marfisi-Schottman et al., 2010). Dans notre expérimentation, le problème d'embrayage multidisque (Yannou et al., 2010) est transformé en un problème où les ressources limitées d'une université sont partagées entre quatre boursiers. Alors que les joueurs négocient pour augmenter leurs bourses, cela revient à la conception collaborative du système d'embrayage multidisque. Les résultats de nos simulations de Monte Carlo et expérimentations du jeu sérieux confirment que l'intensité des conflits est réduite lorsque les conflits sont résolus par notre modèle de gestion des conflits.

Le système de résolution des conflits peut être adopté avec différentes stratégies qui prennent en compte l'architecture de solidarité des agents. Quatre stratégies différentes ont été définies en fonction du degré de promotion de la solidarité. Ces stratégies sont les suivantes, introduites par ordre croissant de solidarité:

- Stratégie 1 : conception non-coopérative. Les agents ne partagent pas d'informations sur leurs états de bien-être et leurs contraintes de sorte que le système de conception n'inclut pas le système de gestion des conflits. Si un conflit de conception se pose, il ne peut être résolu.

- Stratégie 2 : système décentralisé pour la résolution des conflits. Les agents partagent toutes les informations. Si un conflit de conception se pose, les agents sont libres de coopérer en approuvant l'aide, ou de ne pas coopérer en rejetant l'aide. Par conséquent, un agent peut se venger en refusant d'aider l'agent qui ne lui a pas accordé de l'aide lors des étapes précédentes.

- Stratégie 3: système contrôlé pour la résolution des conflits. Les agents partagent toutes les informations. Si un conflit de conception se pose, les agents sont libres de coopérer en approuvant l'aide, ou de ne pas coopérer en rejetant l'aide. Toutefois, si un agent n'approuve pas l'aide, il est pénalisé par un agent de contrôle. Un agent pénalisé ne peut plus définir une contrainte de décision à l'étape suivante du processus où il est disponible pour définir une contrainte. Après la pénalisation, il peut continuer à définir des contraintes de décision. Les agents ne se vengent pas.

- Stratégie 4: système centralisé pour la résolution des conflits. Si un conflit de conception se pose, les agents sont obligés de coopérer en approuvant l'aide.

Les simulations de Monte Carlo qui ont été réalisées avec des agents informatiques montrent que la promotion de la solidarité aide à réduire l'intensité des conflits. Ceci a été confirmé par les expérimentations menées avec des agents humains. Toutefois, ce gain est obtenu au détriment de l'augmentation de la durée du processus. Ceci s'explique par le fait que la résolution des conflits provoque des bouclages lors du processus. En outre, la promotion de la solidarité augmente la satisfaction totale si la frontière de Pareto de l'espace de satisfaction reste convexe tandis que l’on s’approche de la ligne de divergence nulle.

Le modèle résultant de ce travail est nouveau et présente un cadre complet pour améliorer le processus de conception distribuée dans la pratique. Différente de la CAO classique où le dimensionnement n’est pas concourant, notre modèle fournit un mécanisme de dimensionnement concourant par réduction d’incertitude. Cela permettrait également d'éviter les bouclages itératifs, car le modèle de produit ne serait pas représenté par un concept déterministe, mais avec une palette de nombreux concepts : c’est le principe du SBD. Les concepteurs peuvent également évaluer leurs états absolus et relatifs par des indicateurs de bien-être au cours du processus de conception. En les mettant à disposition des autres concepteurs, cela permet de mieux se comprendre et possiblement de s’aider. Dans ce type de CAO distribuée, les conflits de conception pourraient donc être rapidement avérés et résolus.

Chapter 1: Introduction and Motivation

This chapter introduces the general context of the dissertation and its motivations.

1.1 Introduction

1.1.1 Collaborative Distributed Design

The competition in the global market is becoming increasingly intense. Organizations are forced to enhance their competitiveness by responding faster to market changes, providing the market with more innovative technologies and higher quality products, and reducing their product development and production costs. In parallel, products and systems are becoming increasingly complex in order to satisfy demanding market requirements and customer needs. Engineering design is a product development process where a set of functional specifications are transformed through a series of design activities and decisions into a complete description of a physical product or a system that satisfies market requirements (Nahm and Ishikawa, 2004). According to Pahl et al. (2007), this process comprises planning and task clarification, conceptual design, embodiment design, and detail design phases. Embodiment design is the phase where a technical product is dimensioned. Embodiment design problems of complex products and systems are large scale and multidisciplinary. Design decisions for solving these complex design problems require multidisciplinary expertise and multidimensional evaluation of merit and performance. Thus, a single designer cannot provide necessary expertise and rapidity to develop new products in a limited time. Therefore, collaborative design is required in New Product Development (NPD) processes (Favela et al., 1993).

Collaborative design is the involvement of multiple designers from many different disciplines providing necessary expertise and skills together to achieve a common design goal. Each designer plays a different role that engages his/her expertise to the entire process. Designers generate information from their visions and perspectives and influence the design model with their decisions during the design process. Extensive perspectives and different visions are thus also provided to the NPD project, so joint activities of many design actors can obtain better results than that from a single actor. Gray (1990) describes the collaboration as “a process through which parties who see different aspects of a problem can constructively explore their differences and search for solutions that go beyond their own limited vision of what is possible".

There are two system strategies employed for synthesizing collaborations of interacting designers: centralized and decentralized decision-making (Panchal et al., 2007). Centralization in design refers to the collective responsibility, while decentralization refers to the distributed responsibility for making decisions. In the ideal case, centralization would be the simplest system approach for the collaborative solution of engineering problems. However, the centralization of the design system does not represent a realistic application for large scale NPD processes. Firstly, the design problem is significantly complex, so it has multidisciplinary boundaries. A single responsibility does not represent adequate expertise for multidisciplinary decision making. Secondly, current design organizations are becoming increasingly complex and international, so that they have organizational barriers caused by geographical dispersion and temporal differences (Détienne et al., 2004). System decentralization is therefore unavoidable in a large organization dealing with large scale problems (Lee and Whang, 1999).

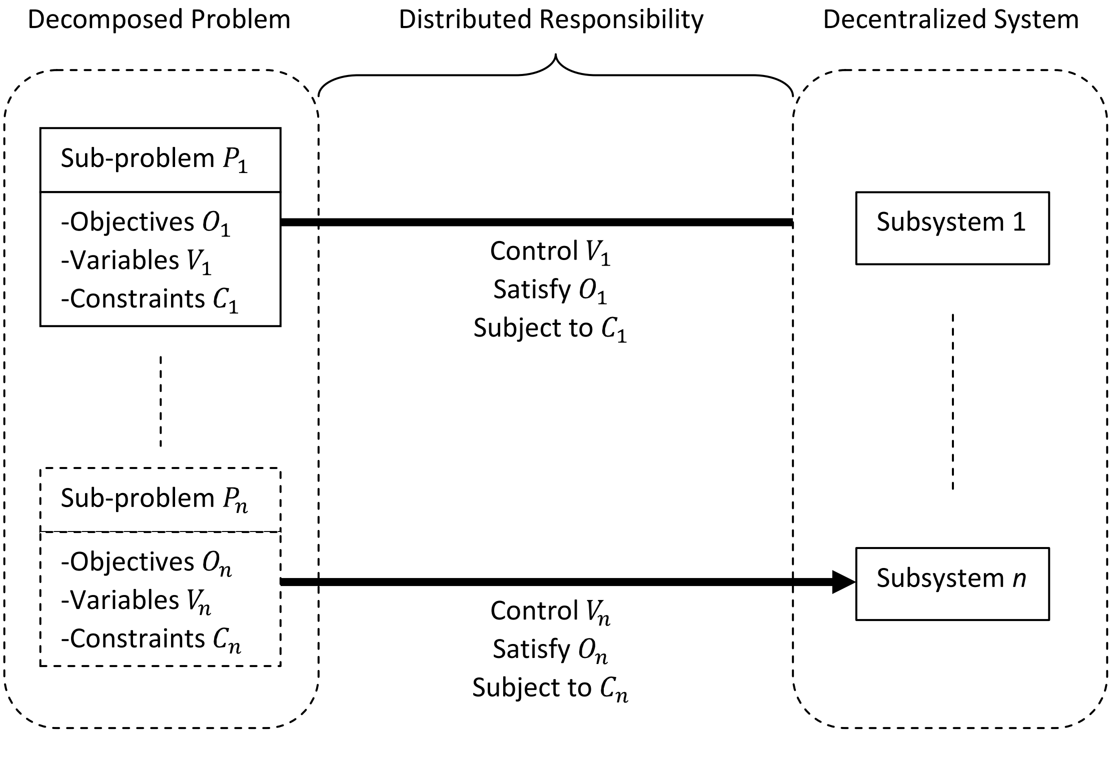


Figure 1.1: Distributed responsibility

Distributed design is a specific form of collaborative design for decentralized systems. In distributed design, the global design problem is decomposed into sub-problems and distributed to subsystems (Papalambros et al., 1997). According to Sobieszczanski-Sobieski et al. (1984), decomposition generally happens in disciplinary boundaries of the multidisciplinary design problem and/or physical divisions between subsystems responsible for the decomposed design problem. Subsystems are stakeholders in the decentralized system, such as design actors and design teams, with limited responsibility. As shown in Figure 1.1, design responsibility is distributed to subsystems. Each subsystem has specific design objectives and limited control over design variables. Local variables V_i and design objectives O_i subject to related problem constraints C_i define their sub-problem P_i.

A distributed design system with each of its subsystems consisting of a single design actor can be viewed as a distributed decision making process of multiple design actors. As shown in Figure 1.2, conventional decision making methods usually consist of the serial execution of design activities of actors (Ceroni and Velásquez, 2003). Design actors make decisions considering their design objectives and define specifications on the design model. M_i is the version of the design model, and S_i is the specifications defined by decisions of design actor i. S_i with M_i produce together the next version of the design model: $S_i + M_i = M_{i+1}$. The modified model M_{i+1} is transferred to the subsequent design actor i+1. If the subsequent design actor is not able to define his/her specifications on the modified model, then this actor sends a feedback to the previous actor for a revision of the design model. This cycle continues until both actors compromise. Next, the model is transferred to another design actor. This is an iterative process where design information is shared sequentially. It often results in long development times, high development costs and low quality implementation. Design actors at the front of the series are more advantageous. They have more freedom to define their specifications, so they influence the model more. However, design actors coming behind have less freedom to define their specifications, because the design model is already heavily modified. Thus, the leaders have to predict what the followers do.

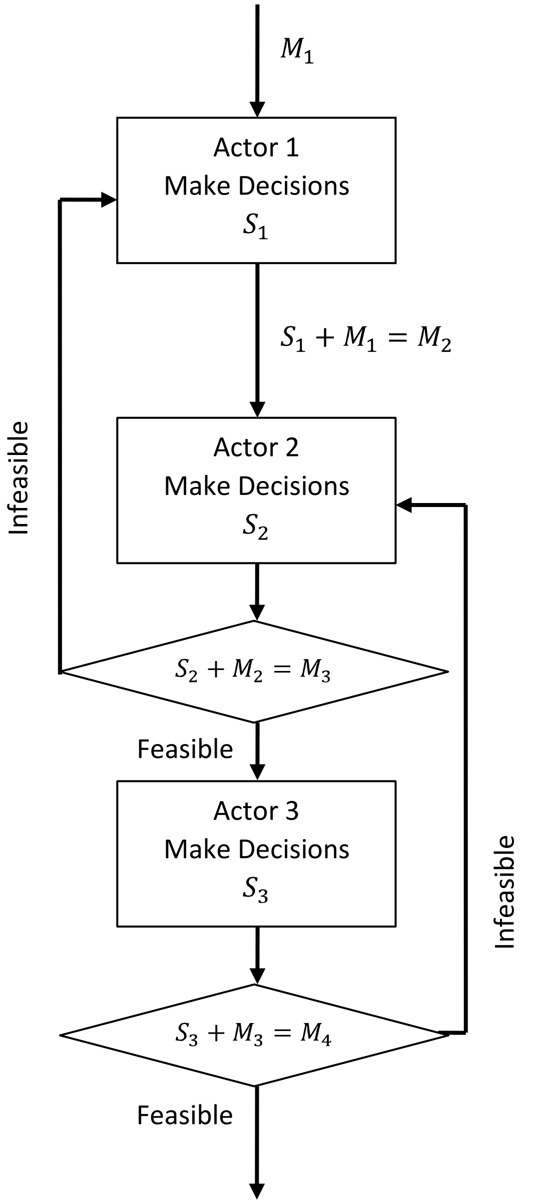


Figure 1.2: Serial decision making in design

The requirements for shorter time, lower cost, and higher quality have turned the design process into the concurrent execution of design activities. Concurrency of the design activities refers to the process architecture where design actors work synchronously. Design actors share design information in parallel. The concurrent decision making in design is demonstrated in Figure 1.3. Each design actor works on the same version of the design model M_i. They have theoretically the same degree of freedom. They define their specifications independently, and an aggregate modification is determined. If $M_i + \sum S_i$ is not feasible than a feedback is sent to every design actor to revise their specifications. With concurrent execution a collaborative compromise is searched, while with the serial execution the solution is the sum of serial individual compromises. Other motivations are reducing the overall design cycle (avoiding expensive design loopbacks for instance), and increasing the design quality.

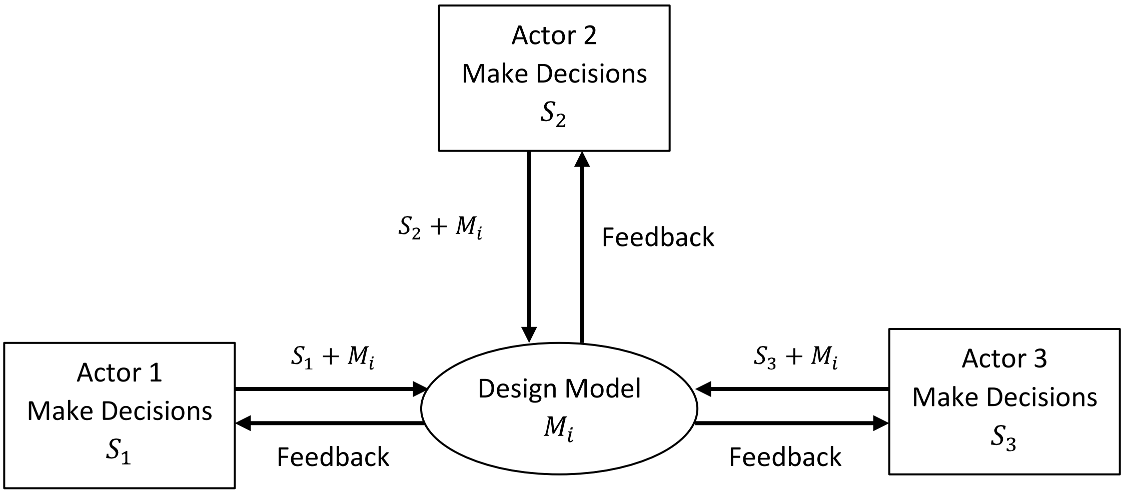


Figure 1.3: Concurrent decision making in design

There are other important motivations as well for the decomposition of the design problem and decentralization of the design system. With decomposition the design problem becomes more manageable, because the complex design problem is divided into simpler sub-problems (Krishnamachari and Papalambros, 1997). Task specialization in the decentralized design system allows designers to work on what they do best, and this provides an efficient use of disciplinary expertise (Harris, 1994). Design capacity can be increased with outsourcing designers working in different time zones. This provides an efficient use of design facilities.

1.1.2 Collaborative Design Problem Solving Methods

There are many methods practiced for the resolution of collaborative design problems. Most of them employ Multi-objective Optimization (MOO), Multidisciplinary Design Optimization (MDO) and Game Theory (GT) approaches. A large scale design problem of a complex product requires multidimensional evaluation of merit and performance. It is typically thus a multi-objective problem. MOO is the most simplistic approach for representing and resolving multi-objective design problems. MOO methods are developed as a centralized decision-making process where different knowledge from various experts is transferred to a single decision-maker. MOO methods can be categorized in two groups; some methods keep the objective functions in a vector form (e.g. ϵ-constraint method), and the others scale objective functions in a single objective function (e.g. utility function method, goal programming method) (Rao and Freiheit, 1991).

Conventional design modeling approaches that adopt MOO assume that there is always a full cooperation in the design process (Jagannatha Rao et al., 1997). Full cooperation refers to mathematical and personal cooperation. In full cooperation, design actors are considered as members of a team, and they interact explicitly in order to improve the entire system solution. The ideal objective is to obtain a collaborative compromise. This is typically a Pareto optimal solution expected from simultaneous considerations of tradeoffs. Full cooperation is the ideal case, it has however some major limitations for complex systems (Ganguly et al., 2008). Full cooperation requires true concurrency where the entire design space is explicit and the information flow among design actors is perfect. According to Lewis and Mistree (1998), it is very difficult to obtain a full cooperation in practice because of the potential information, organization and process obstacles experienced in the design system. It is thus highly unlikely to obtain true concurrency in coupled and complex systems.

MDO explores multidisciplinary interactions of designers, and incorporates various disciplines with frameworks that include some formal definitions for communication and coordination between subsystems of a complex engineered system (Simpson and Martins, 2011). There are various frameworks that include problem resolution approaches in order to guarantee a converging system and an optimal solution. Devendorf and Lewis (2011) list these widely used approaches as follows: target cascading (Kim et al., 2003), concurrent sub-space optimization (Wujek et al., 1996), bilevel integrated system synthesis, and collaborative optimization. MDO has however some major challenges for complex design systems. Devendorf and Lewis (2011) outline the challenges associated with MDO frameworks; such as high communication, coordination, management and organization costs, and difficulty to obtain a global agreement on the proposed framework.

When a decentralized MDO problem does not include any formal framework, it is typically a non-cooperative distributed design problem where design actors act individually. Each design actor has limited knowledge about the design problem. They control local variables unilaterally in order to satisfy their local objectives. Next, they share the information about their local variables. They whether compete and try to improve the model only for their design objectives without considering others' objectives, or there is a leader-follower relationship between the actors where the leader actor dominates the follower actor. Game theoretical approaches can be employed for non-cooperative distributed design problems where the information flow is imperfect (Vincent, 1983). In GT each player constructs a rational reaction set (RRS) which represents player's reactions towards any unknown strategy made by another player. RRSs of constrained problems can be approximately constructed with response surface methodology (Wang, 2003). Although Lewis and Mistree (1998) highlight the extreme difficulty of constructing exact RRSs for complex problems; in their following research (Lewis and Mistree, 2001), they develop some efficient techniques for determining approximate RSSs of complex problems. The intersection of RRSs of the players represents equilibrium solutions called the Nash solution. When players can converge to a Nash solution, they have no incentive to change their solution. A Nash solution is individually stable where neither player has any motivation/reason to unilaterally alter any of the design variables that he/she has control over. However, it is not necessarily a Pareto optimal solution (collectively stable) (Jagannatha Rao et al., 1997). Players can also diverge in an unstable manner (Chanron and Lewis, 2005). System stability criteria are demonstrated by Devendorf and Lewis (2011).

Deterministic methods are considered as point-based approaches where iterative trade-offs are made on point solutions (Liu et al., 2008; Panchal et al., 2007; Sobek et al., 1999). A starting point is determined with multi-attribute targets and adjusted/cascaded iteratively until a stable solution is obtained. The point-based iteration is the classical application of most of the optimization methods (Cooper et al., 2005; Kim et al., 2003) and game theoretical methods (Chanron and Lewis, 2005; Lewis and Mistree, 1998). A point-based iteration may get stuck in a non-Pareto optimal solution or in a local optimum. Alternatively, in set-based design (SBD), trade-offs are made on a solution space derived from problem variables defined as either finite sets if they are discrete, or intervals if they are continuous (Sobek et al., 1999; Ward et al., 1994). The feasible solutions mapped from variable sets/intervals create the solution space. The solution space is thus an n-dimensional Euclidean Space where n is the number of variables. Ranges of a variable represent the minimum and the maximum values that can be assigned to the variable without violating constraints.

SBD is a concurrent engineering design theory where design actors make decisions concurrently by restricting the solution space in parallel considering the ranges of its variables. As shown in Figure 1.4, the remaining solution space is defined by the overlapping decisions. Design actors make further decisions by comparing the remaining solution space to their objective targets. The solution space thus converges to a final solution progressively during design process stages. SBD can be adopted for the technical resolution of design problems with constraint satisfaction problem (CSP) techniques (Meyer and Yvars, 2012; Panchal et al., 2007; Yannou and Harmel, 2006; Yannou et al., 2013). CSP techniques are used as a tool for aiding design where design decisions are formalized through constraints restricting the solution space (Vareilles et al., 2012). With constraint programming (CP), the constraint inconsistency (solution space feasibility) is verified, and the remaining solution space after a design decision is precisely detected (Meyer and Yvars, 2012; Yannou and Harmel, 2004).

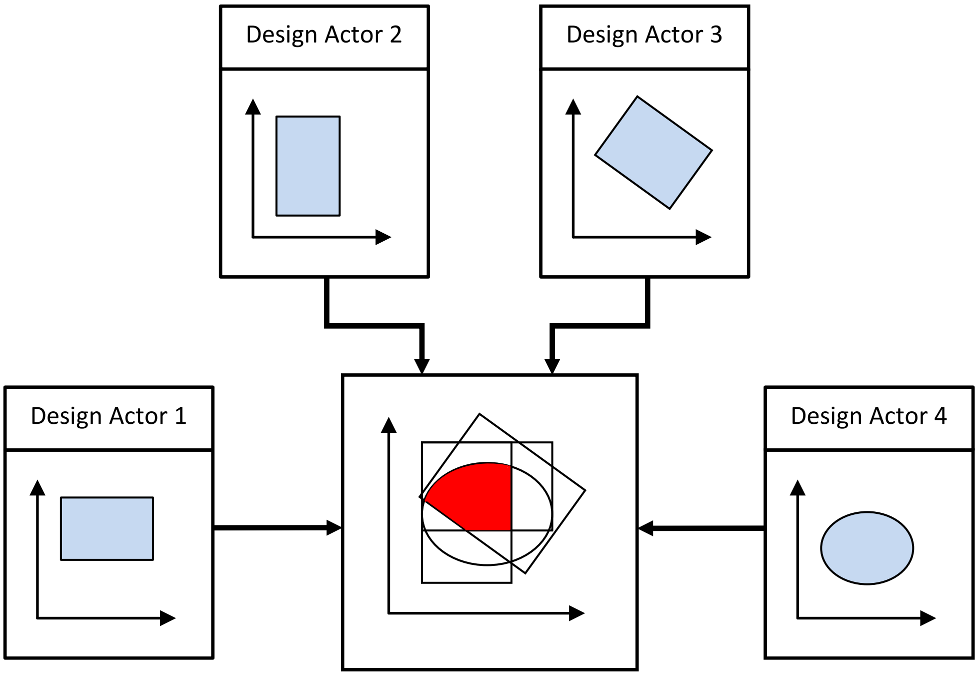
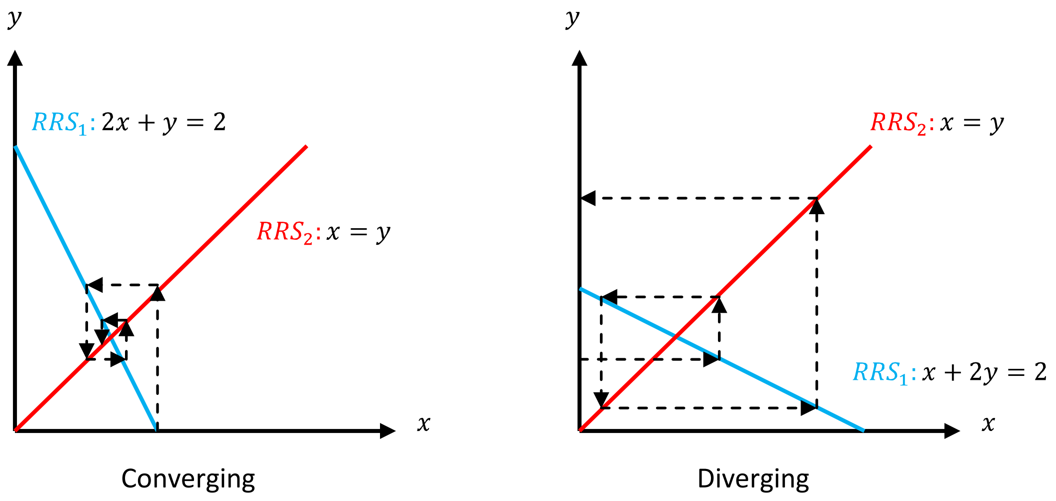


Figure 1.4: Concurrent restriction of the solution space in SBD

Table 1.1: Comparison of Point-based Design and SBD

	Point-based Design	**Set-based Design**
Convergence:	• Iterative on points	• Progressive shrinking
	• Difficult to guarantee	• Guaranteed
Design Freedom:	• Restricted	• Maintained
Design Uncertainty:	• Ignored	• Propagated
		• Reduced systematically

Table 1.1 compares point-based design and set-based design in terms of convergence, design freedom and design uncertainty. As mentioned above, point-based design consists of the iterative convergence of a variable point to a final stable solution. Throughout the iterations, local objective satisfaction levels of different disciplines can alternate while remaining diametrically opposed. This increases the process time or results in divergence. The iterative convergence/divergence in GT is shown in Figure 1.5. With point based design, it is highly difficult to guarantee the convergence stability in complex design problems with strong multidisciplinary interdependencies (Chanron and Lewis, 2005; Klein et al., 2003). The number of iterations that define the process time is thus unpredictable.



Converging

Diverging

Figure 1.5: Iterative convergence/divergence in GT

SBD consists of the progressive convergence of the solution space. The solution space shrinks while further decisions are introduced with constraints during process stages, and converges to an approximate point solution. This is shown in Figure 1.6 where the outer curves represent the solution space, the inner curves represent the specifications defined at the process stage considering the design information emerging from the previous stage, and dashed curves represent the convergence. In SBD, the convergence can be guaranteed by a sequential splitting loop on the intervals of the variables that create the solution space - application of the round-robin strategy (Granvilliers, 2012) as the design process - and rejecting inconsistent constraints with CSP techniques (Yvars, 2010, 2009). While an interval is being split by constraints, its upper and lower bounds approach to each other. This procedure is performed sequentially over all of the intervals while rejecting inconsistent constraints that yield unfeasible solutions. The solution space converges thus to an approximate point solution.

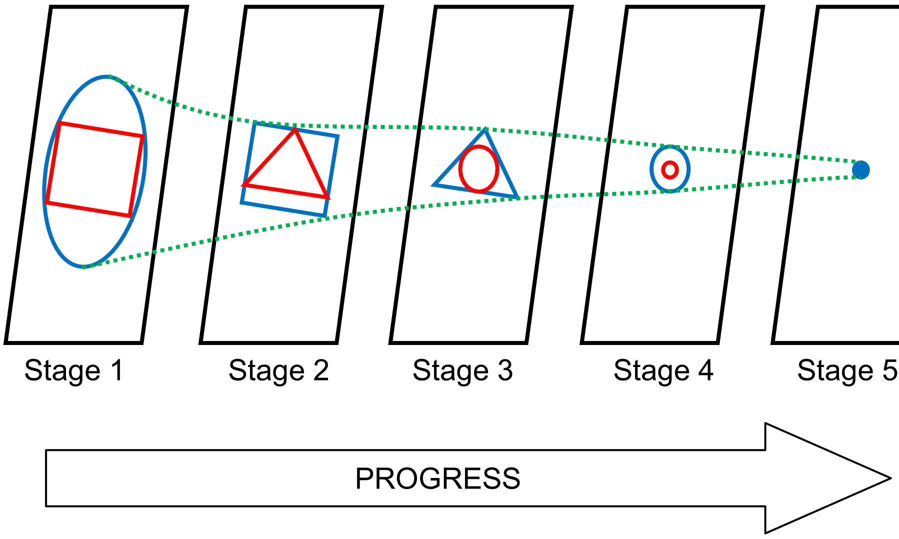


Figure 1.6: Progressive convergence in SBD

The epistemic uncertainty is inherent in design processes (Antonsson and Otto, 1995; Martin and Simpson, 2006; Schlosser and Paredis, 2007). This is caused by the lack of precise knowledge about decision consequences in design. Also, because of couplings, a design actor's decisions are dependent to the unpredictable activities of other design actors. To what value the solution will converge is unknown at the initial state of the design process. If the information about a decision consequence is uncertain, the decision is premature otherwise it is mature. When mature decisions are made, more design information is synthesized. The knowledge about the design thus increases, and some premature decisions become mature. Analysis of the new information allows making further decisions (Yang et al., 2005). This decision-information cycle is shown in Figure 1.7.

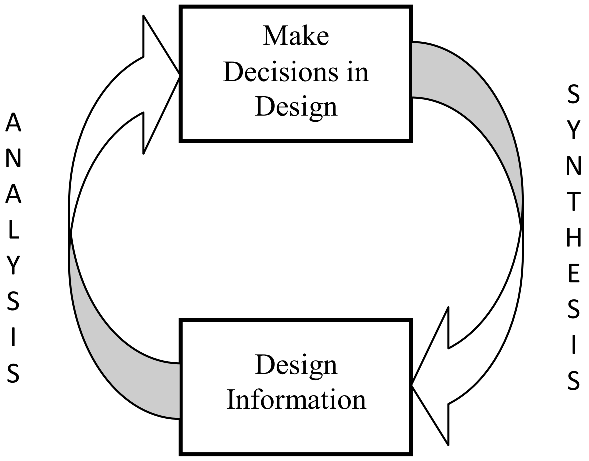


Figure 1.7: Decision-Information cycle

Design freedom is the measure of how the design model is still adjustable while satisfying its design objectives (Simpson et al., 1998). The design freedom is reduced while the design model is restricted with decisions during the design process. Design freedom and design knowledge together represent design flexibility, a measure of the design model's adaptability to changes. Design process time is reduced and design quality is improved with increased flexibility (Simpson et al., 1998; Sobek et al., 1999; Xiao et al., 2005). According to Simpson et al. (1998), the design flexibility of a design system can be improved by increasing the design knowledge while maintaining the design freedom as much as possible. Design freedom is maintained by keeping a set of satisfying alternatives in reserve rather than eliminating all the alternatives to detect only one "best" solution.

The ideal objective of point-based design is to select the best solution alternative, which is the solution that meets point value design targets. The design model must be restricted in a single step in order to obtain a point solution, premature decisions therefore have to be executed in the preliminary design phase. Design uncertainty is therefore largely overlooked, and design freedom is reduced dramatically in a single step. In SBD, the design uncertainty is propagated/represented with ranges of variables showing the worst and best values that can be assigned at an ongoing process (Malak et al., 2009; Schlosser and Paredis, 2007; Yannou, 2004). Design targets are represented by intervals, hence they provide flexibility (Liu et al., 2008; Zhang et al., 2011). The objective is to eliminate the weakest alternatives progressively, instead of trying to find the best alternative directly. Decisions to eliminate the weakest alternatives are made through defining constraints while considering the analyzed design information from the converging solution space. While hard constraints define requirements, soft constraints define preferences about the design model, so that how the design objectives are satisfied can be evaluated (Kelly et al., 2011; Petiot and Dagher, 2011). The execution of a mature design decision increases the design knowledge. Since the emerging design model becomes clearer with more detailed design knowledge, the design uncertainty is reduced. Premature decisions are delayed to subsequent process stages where more design information will be collected. Thus, the design uncertainty is reduced systematically while the design freedom is maintained as much as possible. According to Simpson et al. (1998), better decisions are thus allowed to be made before the freedom to make these decisions is reduced.

1.1.3 Design Conflicts

According to Braha and Maimon (1998), design complexity consists of artifact complexity and design process complexity. They consider two evaluations of complexity for both types: structural design complexity and functional design complexity. Structural design complexity is measured by the multitude of information content in artifacts and design processes. Functional design complexity is the improbability of successfully achieving the required specifications. Similar to this description, Suh (1999) defines the complexity as a measure of uncertainty in achieving a set of specific functions or functional requirements.

In order to manage the structural design complexity in multi-disciplinary design, typically the engineering capability of design systems is increased (Novak and Eppinger, 2001). This means decomposing the design problem into sub-problems and integrating more design actors to deal with these sub-problems. Thus, the structural design complexity is divided into smaller, more manageable sub-problems. However, the requirement of concurrency among subsystems increases. While true concurrency is the ideal case where subsystems generate solutions independently, it is highly difficult to obtain in strongly coupled systems. A coupling is the information that is shared between sub-systems (Wall and Callister, 1995), and it defines interrelations between subsystems. In coupled design, a functional requirement cannot be altered without affecting another functional requirement (Braha and Maimon, 1998). Therefore, the strength of couplings represents functional design complexity (Allison et al., 2006). A coupling pattern is shown in Figure 1.8. Actors have limited control over variables while design performances are influenced by the control variable of the other actor.

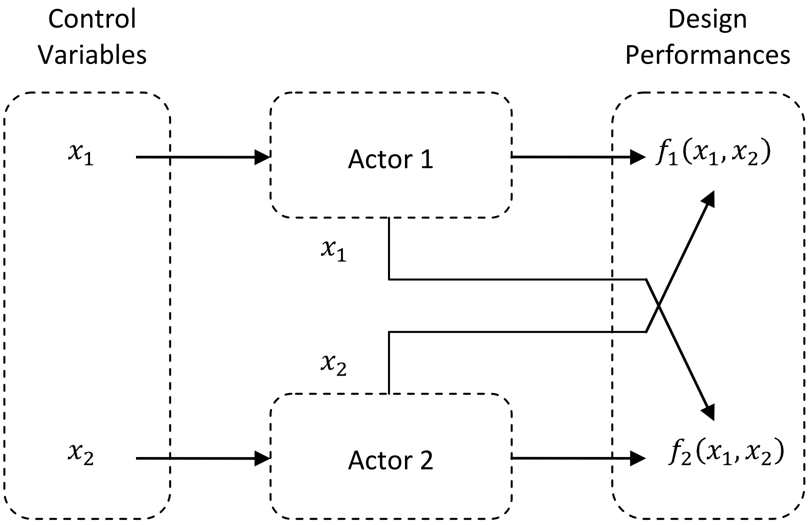


Figure 1.8: Coupling pattern

Obtaining maximum efficiency out of the decomposition is a difficult task because of the presence of couplings among subsystems. Interdisciplinary couplings in multi-disciplinary design represent additional challenges beyond challenges experienced in a single-discipline problem (Sobieszczanski-Sobieski and Haftka, 1997). Sub-problems are carried out by design activities of multiple-actors. Each actor has different skills, visions, perspectives, experiences and expertise. Inconsistencies among their design activities can cause conflicts through couplings. Conflicts are an expression of dissatisfaction or disagreement of interacting stakeholders during the development process of a product, or service (Costantino and Merchant, 1996). Design conflicts can be sorted in two major groups considering their source of inconsistency: problem level conflicts and process level conflicts.

Problem level conflicts: Multidisciplinary problems inherently contain multiple inconsistent objectives. It is impossible to obtain a single solution that fully satisfies all disciplines. While favoring a performance objective of an actor we have to release performance objectives of the other actors. Thus, satisfaction preferences of an actor will cause dissatisfaction on the other actors with conflicting objectives. Figure 1.9 shows an example of problem level inconsistency. s_1 and s_2 are satisfaction levels of two design objectives. The inconsistency -contradiction- of two objectives is represented with the convex curve of the Pareto frontier. s_1^u and s_2^u are ultimate satisfactions that can be obtained separately. While the ultimate satisfaction of an objective is obtained, the other objective is dramatically released. $U(s_1^u, s_2^u)$ is the utopia solution that represents the set of ultimate satisfaction levels of both objectives. Utopia solution is unfeasible since it is out of the Pareto space.

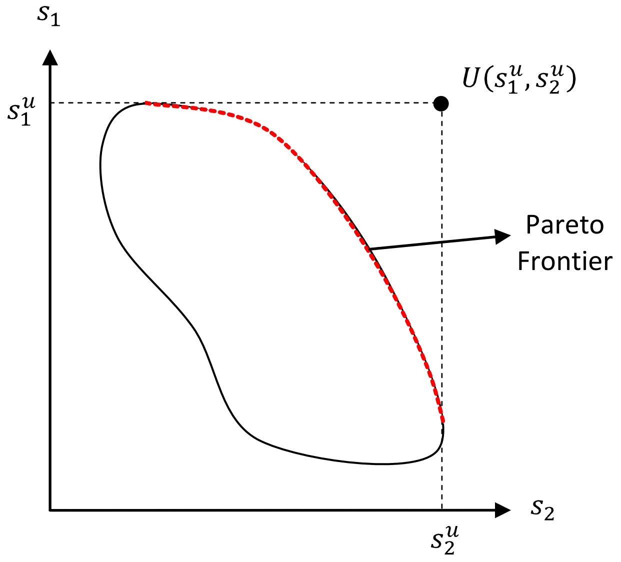


Figure 1.9: Problem level inconsistency

Process level conflicts: In distributed design, design responsibility of a subsystem is limited. Design actors do not have homogeneous control over the Pareto space. Some conflicts thus arise at the process level because of the inexistence or malfunction of the coordination of actors (Brazier et al., 1995; Zhao and Jin, 2003). Process level conflicts are incidences of problem level conflicts. While problem level conflicts are structural and static, process level conflicts are functional and dynamic along the design process. After the decomposition of the problem and distribution of the design activities, individual solutions to sub-problems should be integrated. However, usually individual solutions are inconsistent because design actors tend to prioritize their satisfaction, concentrate only on their particular local problems, dismiss the global problem and overlook collaboration and relationship with the others. Inconsistency of their solutions increases with obstacles arising when they work over distance and in different time zones. Thus, conflicts arise while trying to integrate inconsistent results of individual activities. Figure 1.10 shows an example of process level inconsistency. $specs_1$ and $specs_2$ are specifications defined by two subsystems separately. These specifications constrain the Pareto space. Separately, each one yields a subspace: P_1 and P_2. However, together the result is an unfeasible conflicting solution: $P_1 \cap P_2 = \emptyset$.

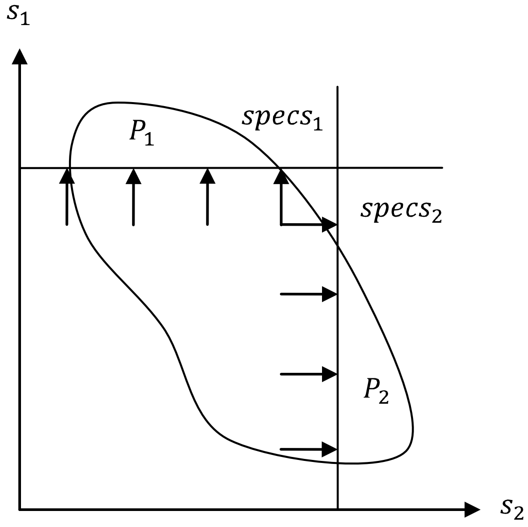


Figure 1.10: Process level inconsistency

Design conflicts are the representation of the divergence, and produce illness during the design process. In order to improve design process performance and design quality, design conflicts must be eliminated systematically as far as the degree of conflict remains low between experts (Jabrouni et al., 2011). A collaborative compromise is therefore approached. The ideal approach is to prevent design conflicts beforehand as much as possible. If some design conflicts could not be avoided, they must be detected and resolved before they become intense. Resolving intense conflicts is very time consuming and destructive on the design model, while it is difficult to detect and justify conflicts before they become intense.

1.2 Research Questions

In SBD, the distributed design process can be performed as a series of process stages where design actors can make decisions sequentially. These decisions are represented as constraints each one yielding a subspace separately. At a process stage, the intersection of subspaces generates the subsequent solution space. This is shown in Figure 1.11 at stages 1 and 3. However, if subspaces do not intersect, together they yield an unfeasible solution space. Since the latter constraint defined at the process stage is inconsistent with former constraints, it is rejected. This is shown in Figure 1.11 at stage 2. The divergence of the solution space is thus avoided, and its convergence is guaranteed. However, satisfaction levels of different design actors can still diverge if the convergence is not collaboratively controlled. The collaborative control means that each design actor can express his/her decisions with the same design freedom.

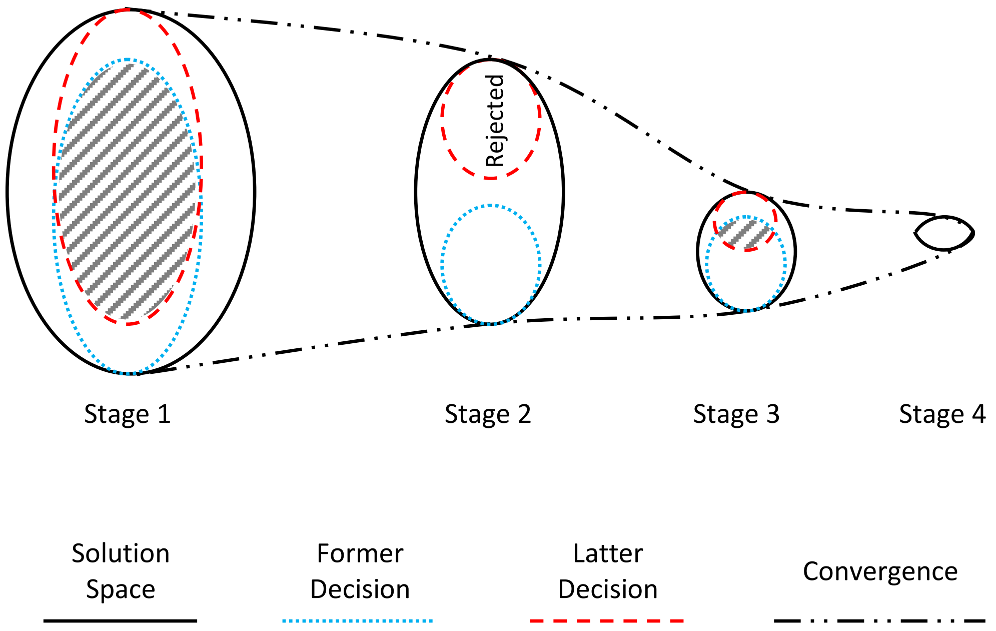


Figure 1.11: Avoiding the divergence of solution space

In a progressive design process, the sequence of the participations of the actors and their speed of progress play a very important role in the shape of convergence. Here, the speed of progress is defined by how frequently, how earlier and how restrictive the decisions are defined. If design actors define their decisions at an earlier stage of the process, they have an advantage. They have more freedom to express their preferences, because they are not forced to follow a bunch of modifications performed previously. Decisions made at earlier process stages thus influence the model more. In contrast, the actors which participate at a later stage have less freedom for satisfying their preferences, they are forced to follow the previous modifications and deal with a restricted design model. If the convergence of the solution space is not controlled collaboratively, the objective function satisfaction levels of different actors diverge. As shown in Figure 1.12, the divergence can increase along the design process. We assume that each rejected constraint represents a potential conflict, because the design decision made by an actor in order to improve the satisfaction level of his/her design objectives is refused. The other assumption is that the conflict intensity would increase while the objective function satisfaction levels of different actors diverge (a design actor is "happy" because his/her objectives are satisfied while another actor is "unhappy" because his/her objectives are not satisfied).

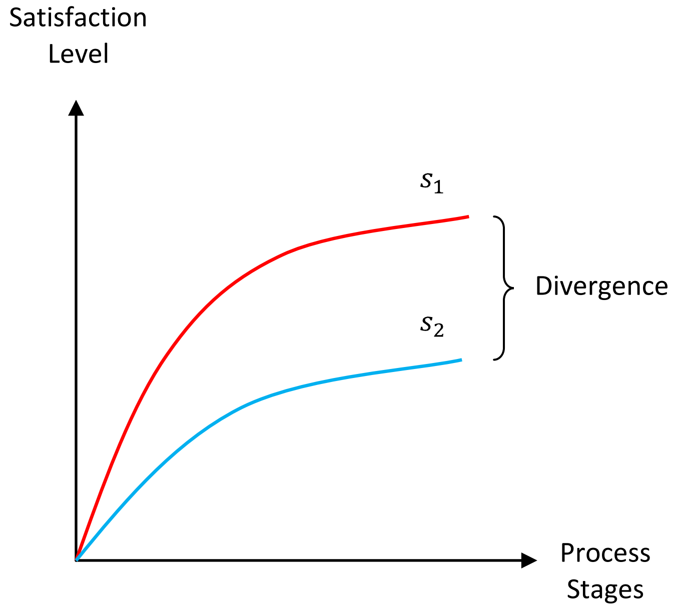


Figure 1.12: Divergence of satisfaction levels

Design process performances; such as the process time, and the global satisfaction (here considered as the total satisfaction); are influenced by how the process is performed. Two main process approaches are top-down and bottom-up design approaches (Fathianathan and Panchal, 2009). In CSP, these approaches correspond to the choice of the space that is controlled. In the top-down design process, design actors control the design space defined by design variables. In the conventional bottom-up design process, design actors control the performance space defined by performance variables. The control space can be heterogeneous in terms of design requirements (Liu et al., 2008). A heterogeneous control space is shown in Figure 1.13. v_1 and v_2 are design variables that can be controlled by two design actors whose satisfaction levels are s_1 and s_2. There is a mapping between the control space and the satisfaction space. The inequality constraints are defined on the control space to eliminate V^1 and V^2, because these solutions give a very low satisfaction (S^1). However, S^2 which is a Pareto optimal solution is lost. Multiple solutions on the control space can be mapped to the same solution on the satisfaction space. Elimination of a solution on the control space does not guarantee the elimination of its corresponding solution on the satisfaction space. Design actors eliminated V^4, but could not eliminate S^3. The result is a low global satisfaction.

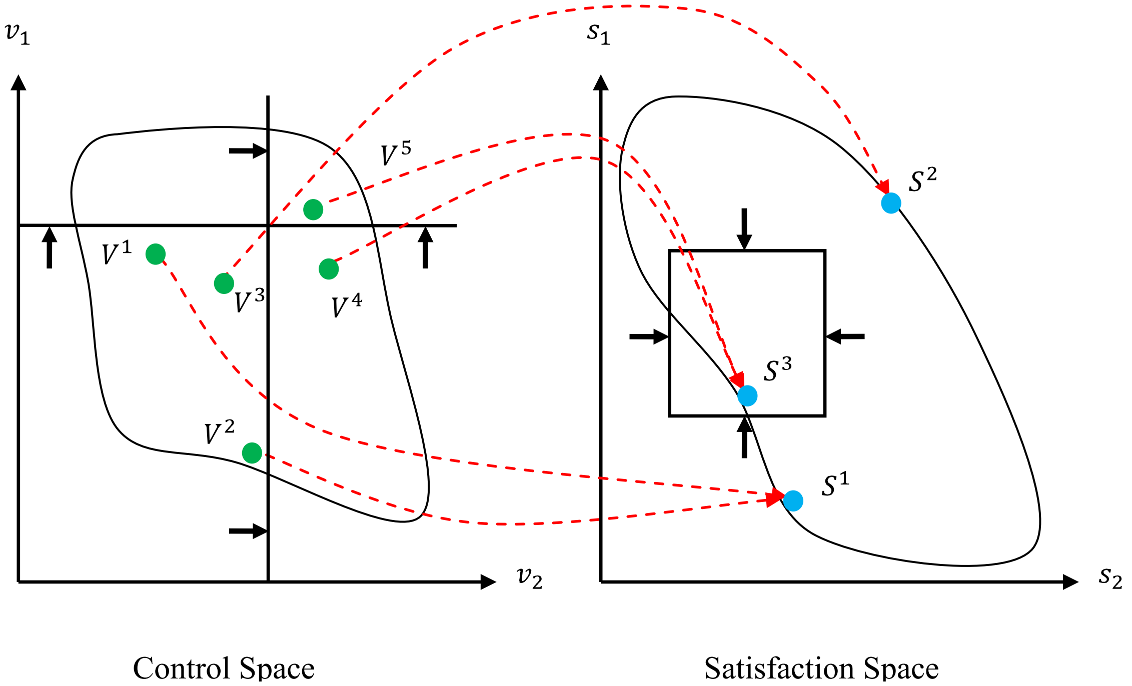


Figure 1.13: Heterogeneous control space

Question 1:

How to prevent design conflicts in distributed SBD while improving process performances?

Human social dynamics can influence design conflicts (Lu et al., 2000; Pelled et al., 1999). Each designer has a unique character. For the same design problem and the same job, different people can react differently. These dynamics are reflected as design attitudes during the design process. In a design team, designers can reflect heterogeneous attitudes. Modeling design attitudes can explore design conflicts. Multi agent system (MAS) is an approach to simulate emerging behavior of the distributed design system.

Question 2:

How to model design attitudes, and prevent conflicts in distributed design systems composed of heterogeneous agents?

If some design conflicts are not avoided, they must be justified and resolved. This requires a conflict management methodology, and an efficient algorithm that allows compromising constraints through relaxing them.

Question 3:

How to justify and resolve design conflicts in distributed SBD?

The conflict resolution system can be adopted for different strategies which take into account the solidarity architecture of design agents of the system participants. A system strategy should be selected to maximize the design performance.

Question 4:

How promoting solidarity is useful in distributed design?

The dissertation is formalized as a series of scientific papers submitted to international journals. Figure 1.14 shows the dissertation organization. Research questions are dealt with in four consequent chapters. Chapters 2, 3 and 4 expose the methodology, the proposed models and their simulations that are conducted by using computer agents. The motivation for developing the models presented in these chapters is to propose a design automation of distributed SBD where design conflicts could be prevented, justified and resolved. In these models, we assume that a design agent is an entity characterized by a number of design attitudes that can influence the design process. These attitudes describe how demanding the design preferences of an agent can be, how restrictively the specifications/constraints are defined by the agent, how fast the agent reacts to reduce the uncertainty, and how the agent is motivated to help for resolving a conflict by compromising some of its own satisfaction. We propose to explore design conflicts by simulating heterogeneous characters through a Monte Carlo approach in order to represent the stochastic reactions in distributed design. Chapter 5 presents the human agent experimentation of the final model emerging from Chapter 4. The motivation for this experimentation is to verify and validate the results of the design automation simulations through human interactions. Human agents are exposed to some strategies that are defined through considering the solidarity architecture in a distributed SBD process.

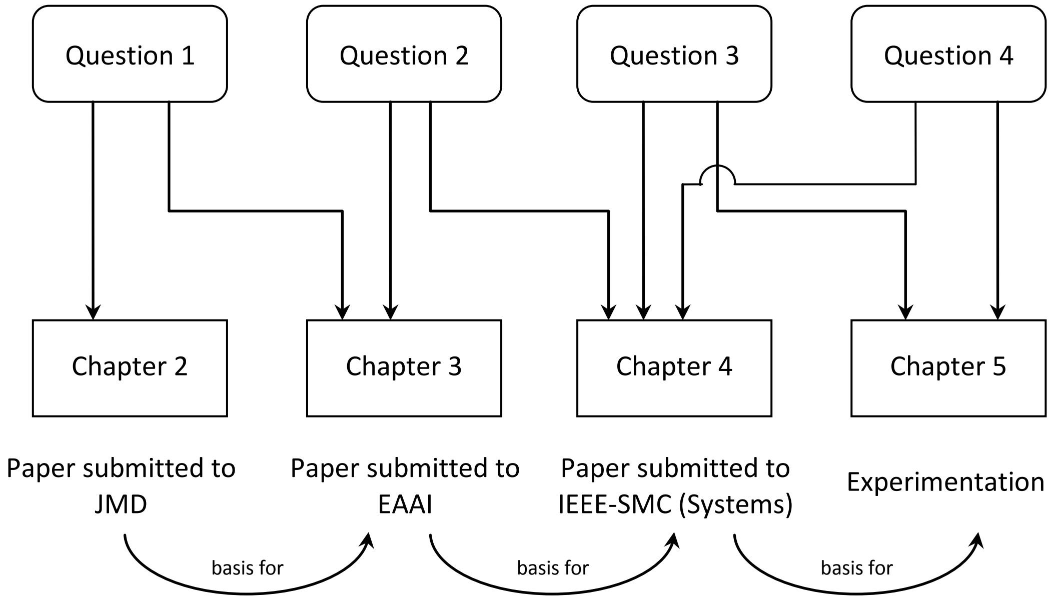


Figure 1.14: Dissertation organization

Chapter 2: Improving Process Performance of Distributed Design Systems with Controlling Wellbeing Indicators of Design Actors

Baris Canbaz, Bernard Yannou, Pierre-Alain Yvars

This paper has been submitted to *Journal of Mechanical Design.*

Abstract

In new complex product development processes, the design problem is usually distributed to multiple actors from different disciplines. Each design actor has a limited responsibility in the design system. Therefore, each design actor has limited control over design variables and performance variables. However, design actors are not isolated since their design activities are coupled. This can generate design conflicts through inconsistencies among design objectives and working procedures. When the design convergence is not controlled, inconsistencies can distort the satisfaction equilibrium between design actors. This means that if a design actor aims at satisfying only his/her local design objective, other actors having conflicting objectives will be dissatisfied. Thus, individual satisfactions diverge. The intensity of conflicts is measured with the satisfaction divergence. In this paper we define wellbeing indicators in order to control the convergence of distributed set-based design (SBD) processes. Wellbeing indicators reflect design actors' satisfaction degree of their process desires. We performed a constraint programming Monte Carlo simulation of our SBD framework with a complex design problem. We compared the results of wellbeing indicators with the results of the processes where design actors do not use wellbeing indicators. It is shown that when design actors have some means to control their convergence, the solution space converges to a solution in satisfaction equilibrium while epistemic uncertainty of the design model is reduced. Some conflicts are therefore prevented and the satisfaction divergence is reduced, leading thus to an improved design process performance.

2.1 Introduction

Collaborative design is the involvement of multiple design actors from different disciplines working together to provide necessary expertise for multi-disciplinary design problems. Distributed design can be performed for collaborative design problems, where design system is decentralized; the global problem is decomposed into sub-problems and distributed to subsystems (Papalambros et al., 1997). Subsystems are composed of design actors or design teams which may be distributed to different geographical locations and even different time zones. Each subsystem has control over some design variables that define the allocated sub-problem and performs concurrent design activities in order to satisfy local design objectives. According to Sobieszczanski-Sobieski et al. (1984) system decomposition generally happens in physical divisions between subsystems and/or disciplinary boundaries of the multi-disciplinary problem. There are important motivating factors of decomposition of the design problem and decentralization of the design system, such as complexity management, decreased development time, efficient use of disciplinary expertise and design facilities, and concurrency of design activities (Balling and Sobieszczanski-Sobieski, 1996; Cramer et al., 1994; Sobieszczanski-Sobieski and Haftka, 1997).

In the ideal case, design actors should be able to work asynchronously and generate solutions to sub-problems independently. True concurrency is thus achieved, and satisfactions of subsystem objectives are maximized equally. However in reality, true concurrency is very difficult to achieve (Lewis and Mistree, 1998), because design actors are not isolated from the other actors' activities. They are related to each other by couplings. Obtaining maximum efficiency out of the decomposition is a difficult task because of the presence of couplings among subsystems. Couplings are shared information and they can cause conflicts through inconsistencies. In a design system, inconsistencies arise at problem level and process level. At problem level, subsystems can have inconsistent perceptions to the same objects. Typically, design actors from different disciplines do not have consistent objectives, but they have conflicting objectives. At process level, design actors tend to have freedom in their working environments and to determine their design strategies for their favor. Thus, their working procedures are optimal in local for their particular sub-problems; however they may be inconsistent with other actors' working procedures and may have negative impacts on the global system solution (Zhao and Jin, 2003).

The divergence of local objective satisfactions of subsystems is the measure of how one subsystem is satisfied more than another. When design objectives of subsystems are not satisfied in equilibrium, such as one subsystem is satisfied causing dissatisfaction to the other, design conflicts arise among subsystems. The divergence represents thus the intensity of design conflicts. In the ideal case where all subsystems are fully satisfied, divergence is equal to zero. Divergent objective satisfaction solutions show that certain participants suffered during the design process, because of not being able to perform their jobs effectively. This represents a low process performance of the design system, because it cannot be a globally satisfactory solution. According to Chanron and Lewis (2005), couplings also generate a challenge for allocating design variables to subsystems. If some design variables influence design performance variables of several subsystems, the allocation technique of design variables is critical, because it can influence the design quality as shown by Kim et al. (2003) and design process performance as shown by Park et al. (2001). In order to obtain a better performance from the design process, objective satisfaction states should converge in equilibrium. However, obtaining satisfaction equilibrium implies a challenge, because design actors do not have means to efficiently control their dissatisfaction from solutions.

Figure 2.1 shows a coupling pattern where the design variables (*a, b*) are related by a constraint but the design actors have limited control. Design actors measure performance variables (*X, Y* and *Z*). Normalized satisfactions of the local subsystem objectives are s_1, s_2 and s_3. The objectives of Actor 1 and Actor 3 are conflicting. When Actor 1 minimizes *a, Z* decreases; Actor 3 is dissatisfied. When Actor 3 maximizes *b, X* increases; Actor 1 is dissatisfied. The objective of Actor 2 generates uncertainty, because the design variables can be both maximized or minimized as far as the objective of *Y* is satisfied. Set-based design (SBD) can overcome the design uncertainty issue by representing uncertain variables with intervals and reducing intervals while collecting design information (Malak et al., 2009). As represented in Figure 2.1, the intervals shrink with specifications shown with dashed arrows, and converge to a point solution $S^C\colon (s_1^C, s_2^C, s_3^C)$. As seen with this example, if design actors can only control design variables, they cannot control their satisfaction and dissatisfaction. This can end with the satisfaction domination of a design actor on another: one objective is satisfied the other is dissatisfied when $Min(a)$ or $Max(b)$. Conflicts become therefore more intense represented with the increasing divergence of actors' individual satisfaction solutions. If the convergence of SBD can be controlled efficiently, this issue can be resolved. With an efficiently controlled design process, a collaborative compromise can be obtained. Thus, satisfaction equilibrium is obtained where individual sub-problem objectives are satisfied as much as possible while their divergence is kept minimal. For the example in Figure 2.1, a collaborative compromise solution S^C: $(s_1^C = 0.5,\ s_2^C = 0.5,\ s_3^C = 0.5)$ can be achieved with $a = 3$ and $b = 15$. Individual satisfactions are equal, so this solution has a zero divergence.

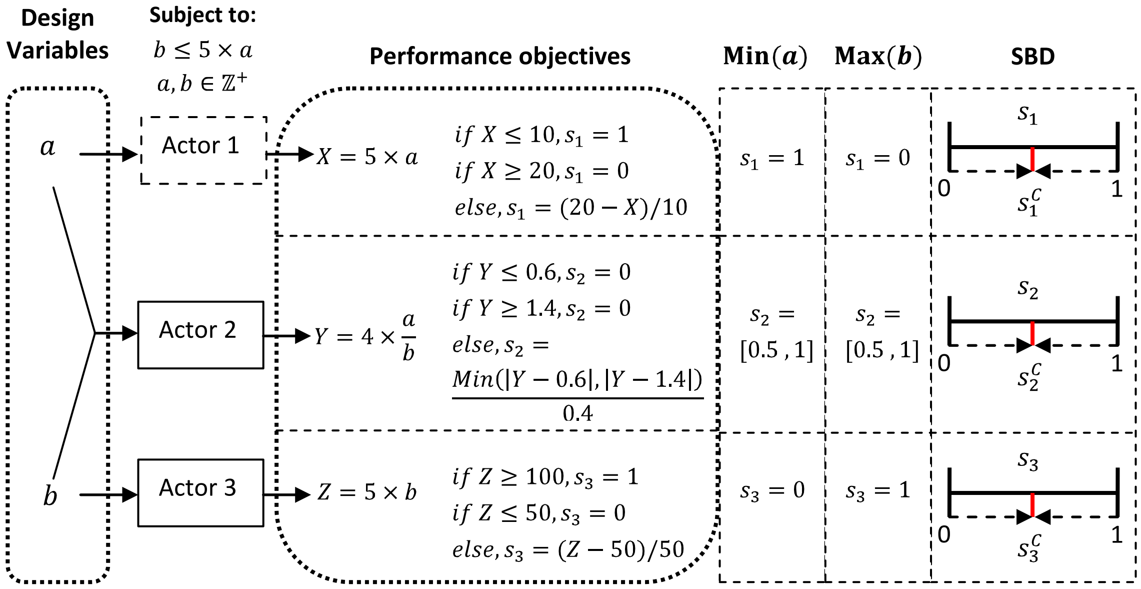


Figure 2.1: Coupled design pattern

In this paper we develop control indicators for SBD, called wellbeing indicators, to provide an alternative to controlling only local variables. Wellbeing indicators are derived from performance variables with product preferences and design actors' process preferences. They allow a bottom-up design process which involves utilizing solutions to identify the values for design parameters while reducing design uncertainty (Fathianathan and Panchal, 2009). They evaluate suffering of design actors during the design process, and provide a controlled convergence in SBD. The objective of this paper is to simulate the design process performance of wellbeing indicators over conventional top-down and bottom-up design approaches. In Section 2.2 the uncertainty management of SBD and Constraint Satisfaction Problem (CSP) techniques are discussed. Our distributed SBD framework is introduced in Section 2.3 and the CSP simulation process of this framework is presented in Section 2.4. Monte Carlo simulations of our approach are performed on a design problem. Problem definitions and simulation results are presented in Section 2.5. The results from Section 2.6 are discussed and further works are identified in Section 2.7.

2.2 Collaborative Design Approaches and SBD

Uncertainty representation and propagation are important issues for design quality in preliminary design (Antonsson and Otto, 1995). This is the epistemic uncertainty of what the problem variable values of the final solution emerging from the convergent design process might be. Deterministic design methods can be considered as point-based optimization approaches, because solutions are usually represented with crisp values, and trade-offs are made on point solutions. In order to achieve an optimum design, these approaches must simplify and restrict the problem so that important uncertainty aspects are overlooked. According to Parsons et al. (1999), point-based optimization approaches usually do not reflect a practical use for the early stage design of complex products, because of the lack of uncertainty propagation ability. Set-Based Design (SBD) is an alternative approach where solutions are represented with feasible regions/intervals of variables (Sobek et al., 1999; Ward et al., 1994). Variable intervals of a sub-problem define its individual solution space. Design activities are executed concurrently and the global solution space is defined by overlapping individual solution spaces of each actor. At the beginning of the design process the solution space is rather large. It is reduced by specifications defined into the model. A convergent solution is therefore obtained at the end of the process where the solution space is reduced at maximum level. Design actors do not know this solution at the initial state but they can only determine it at the end of the process. While the epistemic uncertainty is very high at the preliminary design phase, the convergence of the solution space reduces the epistemic uncertainty and provides adequate information for further design decisions.

At early stages of the design process where the epistemic uncertainty is very high, making direct decisions and searching a single solution can be difficult and inefficient. The uncertainty reduction paradigm of SBD is more efficient in concurrent engineering than point-based approaches. In SBD, instead of negotiating over single solutions, design actors work on a set of alternative solutions. This provides variability of design alternatives and flexibility of modifications in the design process. SBD allows gathering design information before making decisions. If there is not reliable trade-off information concerning a design decision, the decision can be delayed to subsequent process stages where epistemic uncertainty is reduced and more reliable design information is obtained. McKenney et al. (2011) and Wang and Terpenny (2003) show that delaying certain decisions under high epistemic uncertainty can result in higher adaptability to changes at later stages of the design process. Parsons et al. (1999) show also that SBD process provides robustness to design errors. If there is a mistake or a faulty decision in the process, when it is corrected, the solution space can be still wide enough to converge to a solution. Thus, in SBD less time is consumed due to a decrease of repetitive design processes and backtrackings (Sobek et al., 1999).

SBD originated as a management philosophy for concurrent engineering operations. However, Constraint Satisfaction Problem (CSP) techniques can be used to adopt SBD for technical solutions of concurrent engineering problems (Lottaz et al., 2000; Panchal et al., 2007; Yannou et al., 2013; Yvars, 2009). CSP is a concept that is often employed in artificial intelligence, operational research and logic programming (Brailsford et al., 1999). It can be applied in every domain where we look for certain solutions while taking account of many constraints, such as Conceptual Design and Production Planning. CSP is defined by three sets *(V, D, C)* (Montanari, 1974):

$V = \{v_1, \dots, v_n\}$ is the set of variables.

$D = \{d_1, \dots, d_n\}$ is the set of variable domains.

$C = \{c_1, \dots, c_m\}$ is the set of constraints.

Domains *(D)* contain the feasible values of corresponding variables *(V)* that do not violate constraints *(C)*. Constraints are either equalities or inequalities that relate variables to some values or to each other. They are conditional if the restriction requires some conditions to be fulfilled. A complete assignment of the values to the variables, which satisfies all the constraints in the CSP, is called consistent solution. The set of consistent solutions is called solution space. CP techniques perform constraint propagations, interval analysis and branch-and-prune algorithms in order to determine the solution space very quickly for any modification of the design model. CP can handle discrete variables as shown by Montanari (1974) and Mackworth (1977) or continuous variables as shown by Faltings (1994). Dynamic CSP allows adding or removing constraints to the problem model when the problem is not static (Dechter and Dechter, 1988). The problem is altered with the evolving set of constraints. The problem at time stage t is $P^t = \langle P^{t-1}, \Delta \rangle$ where P^{t-1} is the problem defined at the previous stage and $\Delta : P^{t-1} \rightarrow P^t$ is the function of added constraints that maps the previous problem to the problem at stage t. Overviews of the different CSP solving techniques and its application on design problems can be found in works (Lottaz C, 2000; Lottaz et al., 2000; Sam-Haroud and Faltings, 1996; Yannou and Harmel, 2004).

In SBD, domains are represented with intervals, either finite sets or real intervals. The solution space is a subset of the Cartesian product of the intervals since some elements of the Cartesian product can be infeasible considering constraints. In the preliminary design phase where uncertainty is significant, the domains are very large. Through the progress of the design process constraints representing decisions are defined into the model. At any stage of the process designers can benefit from precise and consistent representations of the remaining design space detected by domain reduction/filtering of the CP [23]. The initial domains of the example shown in Figure 2.1 are: $D(a) = [1\,,+\infty)$, $D(b) = [1\,,+\infty)$, $D(X) = [5\,,+\infty)$, $D(Y) = [0.8\,,+\infty)$ and $D(Z) = [5\,,+\infty)$. If a constraint is defined $b \geq 10$, then the inconsistent solutions are filtered, so the domains are reduced: $D(b) = [10\,,+\infty)$. The domain reduction due to the constraint leads to the domain reduction of related parameters: $D(a) = [2\,,+\infty)$, $D(X) = [10, +\infty)$, $D(Y) = [0.8\,,+\infty)$ and $D(Z) = [50\,,+\infty)$. Domain reduction can function with a bottom-up architecture where constraints can be defined directly on value occurrences. For instance, if $X \leq 20$ is added to the problem, the domains are reduced to: $D(a) = [2\,,4]$, $D(b) = [10\,,20]$, $D(X) = [5\,,20]$, $D(Y) = [0.8\,,1.6]$ and $D(Z) = [50\,,100]$. This is a very effective way of representing product specifications in design systems, because it can enable design actors to define constraints directly on their design performance variables and indicators derived from the design performance variables. With its domain reduction ability CP allows modeling and propagating uncertainty on variables and reducing the uncertainty during the design progress. Yannou and Harmel (2004) demonstrate that CP techniques can compete with and outperform fuzzy methods and probabilistic methods on managing uncertainty in design. When designing under uncertainty with CP techniques, the final size of the reduced solution space relative to its initial size shows the relative degrees of freedom of the design (Wood, 2001), a measure that we further use for defining our wellbeing indicators. In collaborative design, degrees of freedom of design actors are bounded because of the couplings in the design system. Current CP approaches contribute towards SBD and include collaborative engineering. They concentrate on the identification of the consistent solutions, but generally they lack control mechanisms which could identify, prevent and resolve design conflicts.

2.3 Controlling Convergence of Distributed SBD

We define a distributed SBD framework where design actors can control their convergence while observing their wellbeing indicators. This framework is derived from our earlier studies in works (Canbaz et al., 2012, 2011). First we define the dynamic sub-problem of a design actor and the dynamic design process in a distributed SBD system. Next satisfaction functions and wellbeing indicators are introduced.

2.3.1 Dynamic Sub-Problem and SBD Process

The design responsibility of a design actor is limited by its sub-problem. Figure 2.2 shows the dynamic sub-problem of the design actor k in a distributed SBD system where the domains of problem variables at process stage t are represented with $D^t(\,)$. V_k represents the set of local design variables over which the actor k has control. V_k' is the set of design variables that are out of the actor's responsibility. Thus, the actor k can modify V_k , while V_k' can only be observed. Since sub-problems are coupled, some design variables can be shared and modifiable by several design actors. Design performance variables evaluate the designed product considering design objectives. The design objectives derived from the product preferences coming from the market specifications. P_k represents the set of local performance variables related directly to the responsibility of actor k , and corresponds to the local objectives. They are a function of V_k. Global performance variable (P_G) which is a function of V_k and V_k' , evaluates the global system solution, and corresponds to the global objective shared by all design actors. Design performance variables can be derived from design variables (e.g. weight of an object) as well as a design variable can be directly a design performance variable (e.g. length of an object). The design actor makes decisions on the design model considering the design information about how the design objectives are satisfied by the design performance variables.

A set of initial constraints (C_i) is defined at the initial state of the design model in order to ensure the feasibility of the product. A set of decision constraints (C_d) is introduced by actor k during the design process to make decisions in order to satisfy design objectives while reducing the epistemic uncertainty. If the information for making a decision is uncertain because of the dense couplings among sub-problems, the decision is delayed to a process stage where the epistemic uncertainty is reduced by preceding decisions of the same design actor or the other actors. This is a progressive process where a decision constraint generates adequate information that allows making further decisions. Thus, C_d evolves through subsequent process stages. Domains of the sub-problem (D^t) are therefore restricted progressively at each process stage *t*. The sub-problem of actor k is restricted progressively also by an evolving set of decision constraints C_d' introduced by actors with couplings. With the increasing number of decision constraints introduced to the problem, the sub-problem is dynamic through the design process stages. The aggregated sub-problems propagate the dynamic design model that converges to a narrower solution space continuously.

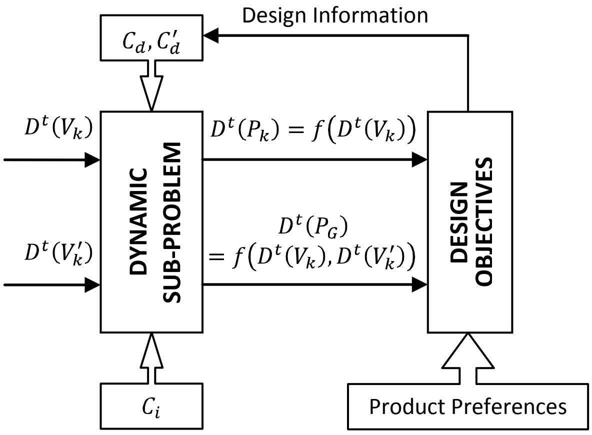


Figure 2.2: Dynamic sub-problem

The dynamic design process is shown in Figure 2.3. In a design process stage, there are three decisions made: D1, D2 and D3. D1 and D3 are Boolean decisions and D2 is a "how" decision. D1 determines whether the actor compromises or defines constraints. Design actors compromise when their design objectives are sufficiently satisfied. Otherwise, they define decision constraints to satisfy their objectives until they compromise. D2 determines how restrictive decision constraints are defined. After the definition of new constraints the feasibility of the design model is tested. D3 determines whether the model accepts the modification or refuses it. The model modified with a decision constraint is accepted if there is at least one feasible solution. If there is not any feasible solution after the definition of a decision constraint, then the constraint is rejected. This can lead to a potential conflict between design actors, because the rejection of the constraint can yield to unsatisfied objectives. This conflicting situation is highlighted with the dashed line in Figure 2.3. Variable intervals shrink with the accepted modifications. Updated variable intervals and the acceptance or the rejection of the constraint of a design actor are emerging design information of the process stage. Design uncertainty is reduced with this information. At the subsequent design stage, actors make decisions considering the design information emerged from previous design stages. At a process stage, D1, D2 and D3 therefore depend on previous decisions.

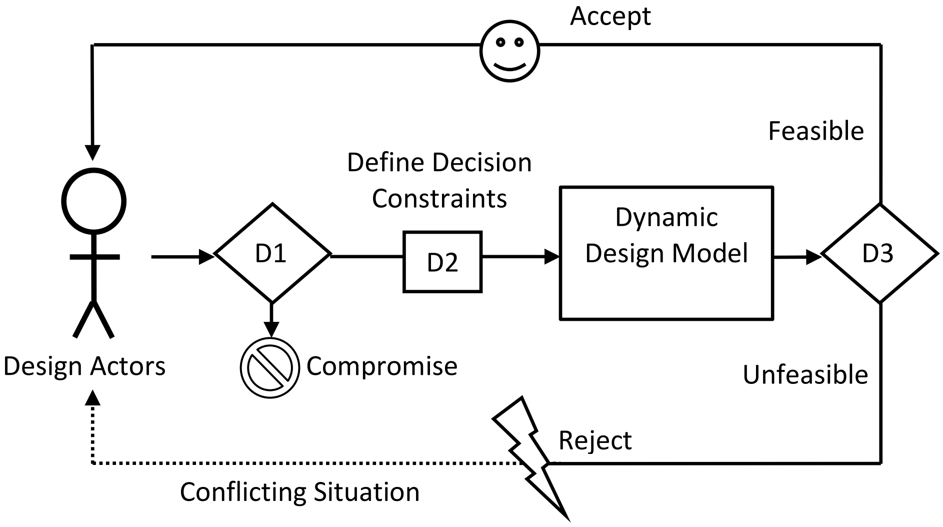


Figure 2.3: Dynamic SBD process

Decisions performed at the design stage also depend on the attitudes of design actors. D1 and D2 are individual decisions and depend on how restrictive the design attitude of the processing design actor is. Design actors with more restrictive attitudes compromise at a higher satisfaction level and define more restrictive decision constraints. D3 is a collaborative attitude that depends on how the design model is restricted by all design actors at previous stages and how D2 is performed by the processing actor. If the model is restricted too much at previous stages, constraints can be rejected even if D2 is not very restrictive. If D2 is very restrictive, constraints can be rejected even if the model is not restricted too much at previous stages. D1, D2 and D3 are explained in subsequent sections.

2.3.2 Wellbeing Indicators

Design objectives are defined as a function of product preferences of the market. A performance variable is evaluated by preference statements to determine how its design objective is satisfied. We combine soft and hard feasibility preference functions of physical programming defined by Messac (1996). Hard feasibility preference statements are as follows:

S1: fully **satisfied** by a performance variable **above** a certain value.

S2: fully **satisfied** by a performance variable **below** a certain value.

S3: fully **satisfied** by a performance variable **equal** to a certain value.

S4: fully **satisfied** by a performance variable **between** certain values.

S5: fully **dissatisfied** by a performance variable **above** a certain value.

S6: fully **dissatisfied** by a performance variable **below** a certain value.

S7: fully **dissatisfied** by a performance variable **equal** to a certain value.

S8: fully **dissatisfied** by a performance variable **between** certain values.

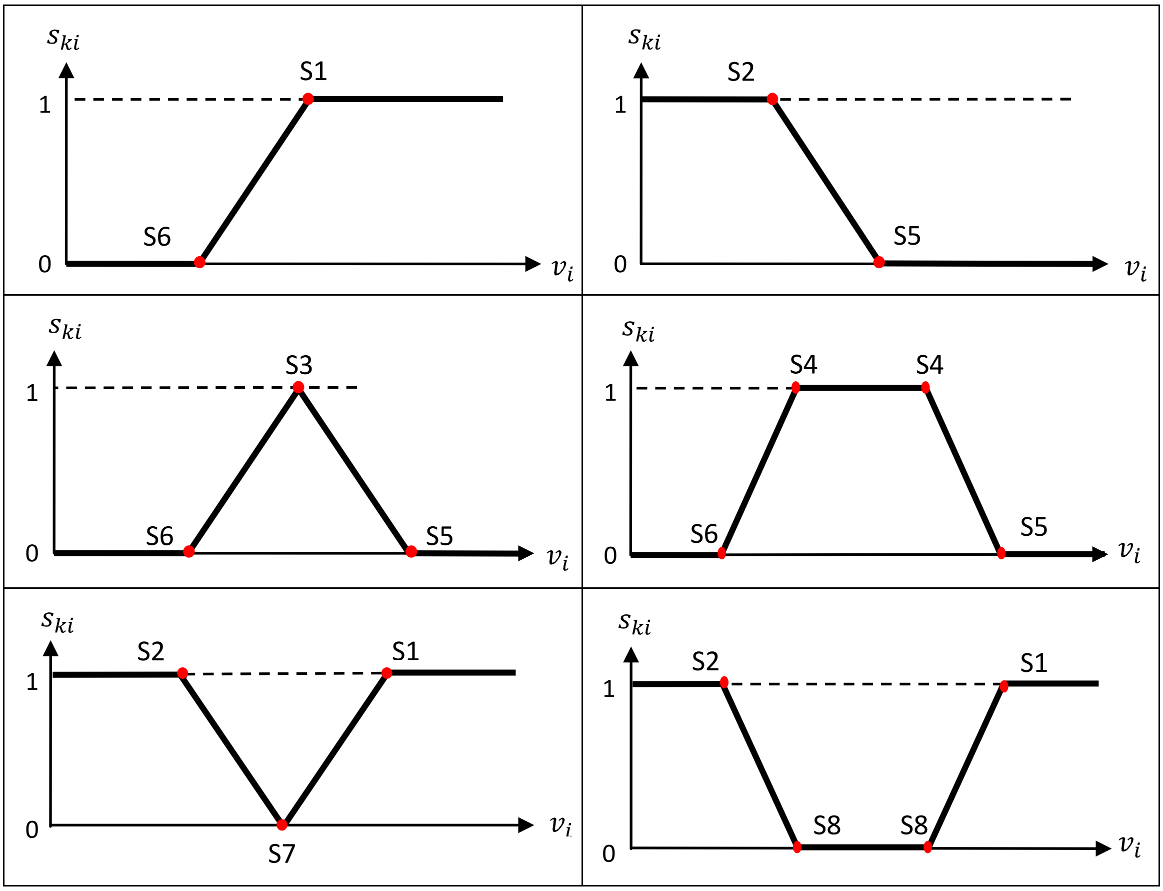


Figure 2.4: Satisfaction functions

Satisfaction values of design objectives reflect product satisfactions of related design actors. With preference statements segmented satisfaction functions are defined. $s_{ki} = f_s(v_i)$ where s_{ki} is the satisfaction value of the design actor k by the performance variable i and v_i is the value of the design performance variable i. Hard feasibility preferences are: $s_{ki} = 1$ if the objective is fully satisfied, and $s_{ki} = 0$ if the objective is fully dissatisfied. Soft feasibility preferences are the transitions between fully satisfied and fully dissatisfied states: $1 > s_{ki} > 0$. In this paper we assume that transitions are linear functions, however nonlinear transitions can be applied with the same definitions. Figure 2.4 represents all the different satisfaction functions derived from the preference statements listed above. We integrate piecewise constraints reflecting design performance variable preferences into the model in order to determine satisfaction states of the design actors. These piecewise constraints define additional information into the model without eliminating any part of the solution space. For example an objective of a design actor k is minimizing a performance variable i; the design actor is fully satisfied by a performance variable value below or equal to $pref1_{ki}$ and fully dissatisfied by a performance variable value above or equal to $pref2_{ki}$. Then the piecewise constraints integrated into the model are:

If $v_i \leq pref1_{ki}\, , s_{ki} = 1$ (2.1)

If $v_i \geq pref2_{ki}\, , s_{ki} = 0$ (2.2)

If $pref1_{ki} < v_i < pref2_{ki}\, , 1 > s_{ki} > 0$ (Linear) (2.3)

In SBD framework we obtain an interval value for performance variables because design variables are defined with intervals. The intervals are reduced through the decision constraints defined during process stages. At process stage t, performance variable i has a minimum value x_i^t and a maximum value y_i^t, the interval of the performance variable i at process stage t is: $[x_i^t\, , y_i^t]$. Since the performance variable is defined with an interval we obtain an interval for the satisfaction of the design actor k by the performance variable i at stage t: $s_{ki} = [mins_{ki}^t\, , maxs_{ki}^t]$ where $mins_{ki}^t$ is the minimum satisfaction and $maxs_{ki}^t$ is the maximum satisfaction obtained within the interval $[x_i^t\, , y_i^t]$. Figure 2.5 represents an example. Piecewise constraints are same as above. The minimum satisfaction is obtained at point A and the maximum satisfaction is obtained between point B and $pref1_{ki}$. During the progress while uncertainty is reduced design actors can observe the potential maximum and minimum satisfaction values from design performance variables.

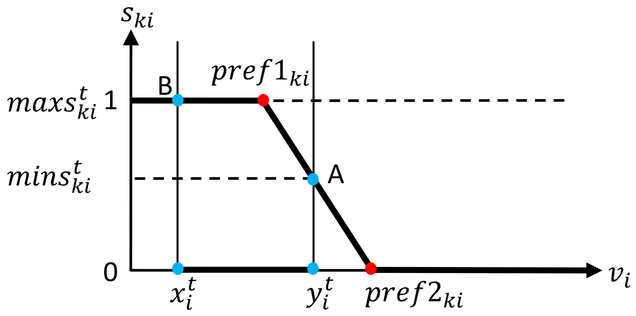


Figure 2.5: Intervals on the satisfaction function

Design actors can assign weights to their satisfaction values obtained from individual performance variables, regarding the importance of the design performance variables for their job. This can be performed if the sub-problem is scalable. Otherwise the sub-problem should be decomposed and distributed to another design actor. In order to observe general satisfaction states of design actors, individual performance objective satisfaction values are aggregated. Design actors can still define constraints on their individual performance objective satisfaction intervals, although they are aggregated. General satisfaction of the design actor k from the whole design model at stage t is the sum of the design actor's weighted satisfactions from all performance variables i at stage t. It is defined as an interval $s_k = [mins_k^t\ , maxs_k^t]$. w_{ki} is the weight assigned to the performance variable i by the design actor k. I is the total number of the performance variables considered by design actor k. Thus satisfaction indicators (Minimum and Maximum Satisfaction) of a design actor are calculated as following equations:

$$mins_k^t = \sum_{i=1}^{I} w_{ki} \times mins_{ki}^t \qquad (2.4)$$

$$maxs_k^t = \sum_{i=1}^{I} w_{ki} \times maxs_{ki}^t \qquad (2.5)$$

$$\sum_{i=1}^{I} w_{ki} = 1 \quad \forall k \qquad (2.6)$$

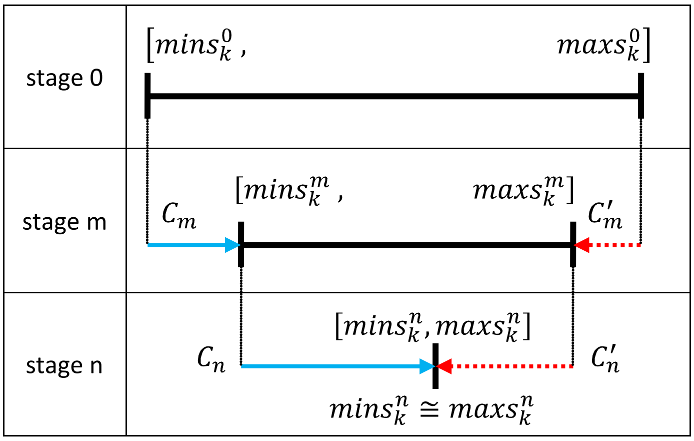


Figure 2.6: Bilateral convergence of satisfaction interval

The satisfaction interval of an actor converges with the progress of the model during the design process stages where more decision constraints are added into the model. Convergence is usually bilateral, because design activities are coupled and conflicting.

The minimum bound of the satisfaction interval is increased by the design activities of its design actor however the maximum bound of the interval is reduced by coupled and conflicting design activities of other design actors. Thus design actors' degree of freedom is bounded. At the final stage of the design process satisfaction interval converges to a solution where Minimum and Maximum Satisfaction Indicators are approximately equal. Figure 2.6 explains the bilateral convergence. C_m and C_n are the sets of constraints added to the problem by actor k increasing $mins_k$. C'_m and C'_n are the sets of constraints added to the problem by the other design actors with conflicting objectives decreasing $maxs_k$.

In a coupled design system, it is typically impossible to fully satisfy all design objectives, because the convergence is bilateral. Design actors are forced to compromise at a certain satisfaction level in the design process. A design actor can define a preference about his/her satisfaction value in which he/she may compromise. This is defined by considering solution space, design uncertainty and also the other actors' degree of freedom. This preference is the compromise threshold value (T_k). T_k represents the satisfaction value that design actor k wants to guarantee in the s_k interval. If $mins_k^t < T_k$, then the design actor defines decision constraints in order to improve s_k considering T_k. If the minimum satisfaction of a design actor by the model reaches T_k value or passes beyond, then the design actor passes to the compromise state. In the compromise state design actors stop adding decision constraints to the model. This leaves space to the other design actors, because maximum values of their satisfaction intervals are not restricted by actors in compromise state. T_k value defines the design attitude that determines D2 in Figure 2.3.

T_k is a process preference of the design actor, different than product preferences. While product preferences define satisfaction functions that evaluate the feasibility satisfaction of the product, T_k values are actors' compromise desires that evaluate process satisfactions of design actors. In order to make a distinction from the design objective satisfaction (s_k), we call the process satisfaction as "wellbeing" of design actor. As shown in Eq. (2.7), s_k is normalized by T_k , and this provides the wellbeing indicator of actor k (wb_k). Figure 2.7 shows how wb_k is derived by using the example explained with Eqs. (2.1-2.3) and considering that v_i is the only performance variable measured by actor k. Wellbeing is represented with an interval

$wb_k = [minwb_k^t, maxwb_k^t]$ where the minimum value is the minimum wellbeing indicator and the maximum value is the maximum wellbeing Indicator. Wellbeing interval converges through the progress of the design process. If the convergent wellbeing value is larger than or equal to 1 then the design actor is in a perfect wellbeing state. The worst wellbeing state is when the value is equal to 0. The convergent wb_k is shown in Eq. (2.8) where dv_k^- is the underachievement deviation variable, and dv_k^+ is the overachievement deviation variable. The process objective of actor k is to minimize dv_k^-. Wellbeing states reveal if a design actor suffers because of not being able to approach to the compromise state, or if a design actor could have the freedom to perform modifications to the model and could have approached to the compromise state.

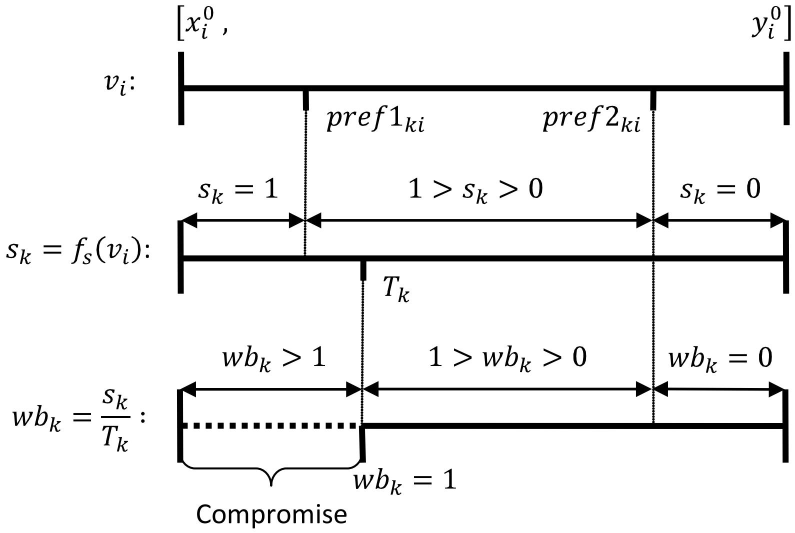


Figure 2.7: Derivation of wellbeing indicator

$$wb_k = \frac{s_k}{T_k} \tag{2.7}$$

$$wb_k + dv_k^- - dv_k^+ = 1 \tag{2.8}$$

$$dv_k^-, dv_k^+ \geq 0$$

$$dv_k^- \times dv_k^+ = 0$$

2.4 CP Simulation Process

We present a CSP simulation of our SBD framework. The objective is to simulate the process performance of a wellbeing controlled design scenario compared to some scenarios that represent general design practices of top-down and bottom-up design. In top-down design practice, design actors usually modify only their design variables while evaluating their performance variables. The process can be performed with modifying one design variable of an actor after one design variable of another actor or with an all-at-once approach where modifications are performed on all the design variables of an actor after all the design variables of another actor. Alternatively, with CSP techniques a bottom-up design approach can be adopted. Design actors can modify their performance variables or wellbeing indicators derived from the performance variables. We define four simulation cases which represent these scenarios. Case 1 and Case 2 represent conventional top-down design processes while Case 3 is a conventional bottom-up process and Case 4 is wellbeing controlled bottom-up design process.

- Case 1: Players define decision constraints on their normalized local design variables. Each player can define maximum one constraint per iteration.
- Case 2: Players define decision constraints on their normalized local design variables with an all-at-once approach.
- Case 3: Players define decision constraints on their normalized performance variables.
- Case 4: Players define constraints on their wellbeing indicators.

The simulation algorithm is shown in Figure 2.8. For the simulation we call design actors as players and process stages as iterations. In the simulation process we used a split mechanism similar to the round-robin strategy that loops on all the variables at process iteration (Granvilliers, 2012). The objective is to ensure a global system convergence where the upper value and the lower value are as close as possible for each variable interval. Intervals are reduced until a good degree of precision is obtained. We make some assumptions for players defining decision constraints:

- If $mins_k^t < T_k$, each player k can define (a) decision constraint(s) only once at any iteration and constraints are defined sequentially. If all the players are processed in iteration then the process passes to the next iteration: t++.
- Decision constraints are defined for improving the worst case scenarios with a coefficient of restriction $M_k > 0$ or $M_{ki} > 0$ or $M_{kj} > 0 \quad \forall k, i, j$. This coefficient is the design attitude of player k that determines how the decision constraint is defined. This is the D3 shown in Figure 2.3. Initial worst cases are larger than 0: $minVn_j^0 > 0\ \forall j$, $minPn_i^0 > 0\ \forall i$, $minwb_k^0 > 0\ \forall k$. If $mins_k^t \geq T_k$ then the compromising player is extracted from the splitting loop (Cases 1 and 2: $M_{kj} = 0\ \forall j$, Case 3: $M_{ki} = 0\ \forall i$, Case 4: $M_k = 0$).
    - Cases 1 and 2: $Vn_j \geq minVn_j^t \times (1 + M_{kj})$ where Vn_j is the normalized design variable *j*, $minVn_j^t$ is its minimum value at iteration *t* and M_{kj} is the coefficient of restriction for Vn_j defined by player *k*.
    - Case 3: $Pn_i \geq minPn_i^t \times (1 + M_{ki})$ where Pn_i is the normalized performance variable *i*, $minPn_i^t$ is its minimum value at iteration *t* and M_{ki} is the coefficient of restriction for Pn_i defined by player *k*.
    - Case 4: $wb_k \geq minwb_k^t \times (1 + M_k)$.
- If a constraint is unfeasible, it is rejected. Then, its related coefficient of restriction value is reduced by half. If the coefficient of restriction value of a variable reaches a precision value (*P*), then the splitting is stopped for this variable, because the upper and lower bounds of its interval are as close as possible considering *P*. If all the coefficient of restriction values reach *P*, then the simulation process stops.
- T_k and M_k are the attitudes of players and defined before the process starts. M_{ki} and M_{kj} values are equal to M_k at the initial state. Product preferences and T_k values do not change during the design process, because they represent static desires.

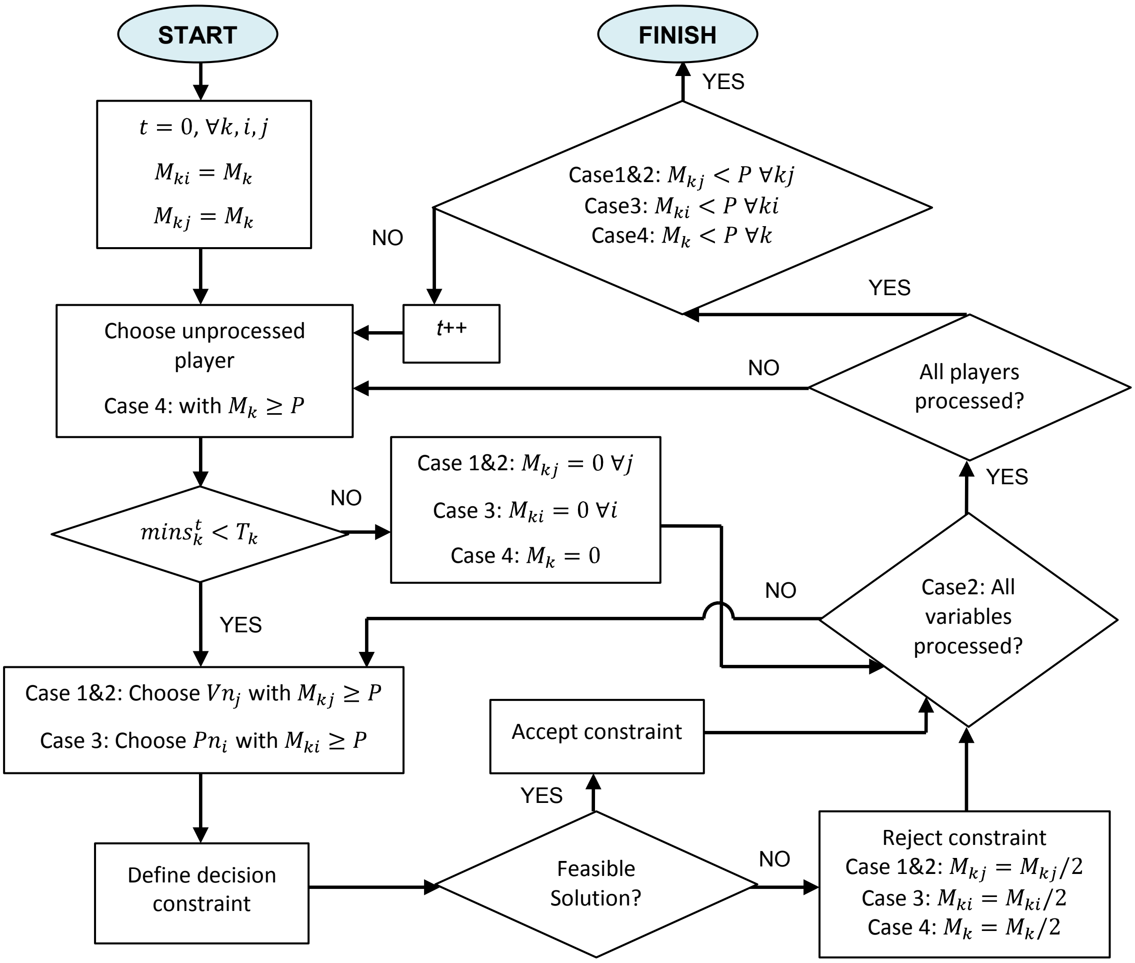

START
FINISH
YES
$t = 0, \forall k, i, j$
$M_{ki} = M_k$
$M_{kj} = M_k$
Case1&2: $M_{kj} < P \ \forall kj$
Case3: $M_{ki} < P \ \forall ki$
Case4: $M_k < P \ \forall k$
NO
YES
Choose unprocessed player
Case 4: with $M_k \geq P$
t++
NO
All players processed?
$mins_k^t < T_k$
NO
Case 1&2: $M_{kj} = 0 \ \forall j$
Case 3: $M_{ki} = 0 \ \forall i$
Case 4: $M_k = 0$
YES
NO
Case2: All variables processed?
YES
Case 1&2: Choose Vn_j with $M_{kj} \geq P$
Case 3: Choose Pn_i with $M_{ki} \geq P$
Accept constraint
YES
Define decision constraint
Feasible Solution?
NO
Reject constraint
Case 1&2: $M_{kj} = M_{kj}/2$
Case 3: $M_{ki} = M_{ki}/2$
Case 4: $M_k = M_k/2$

Figure 2.8: Simulation algorithm

2.4.1 Process Performance Criteria

The simulation process is evaluated by four process performance criteria: number of iterations, number of failures, global objective satisfaction and satisfaction divergence of individual solutions. Four simulation cases are compared regarding these criteria.

Number of iterations and number of failures

A smaller number of iterations represents a faster convergence and a rapid design process. The process rapidity should be evaluated with the number of failures. When a decision constraint is rejected, it is a process failure. Each failure is a potential conflict among players, because the rejection of a decision may be caused by earlier decisions of a player with a conflicting objective. Therefore, less failures means a less conflicting design convergence. The total number of failures represents the number of conflicts occurring in a design process. When there are fewer failures, the coefficient of restriction is split at later iteration, which leads to the number of iterations increasing.

Global Satisfaction and Satisfaction Divergence

Players' local objective is to minimize dv_k^- in Eq. (2.8). In the ideal case, players should obtain the same s_k value and each wb_k value should be larger than 1. The satisfaction divergence is derived from the absolute differences of players' s_k values. All d_i values calculated by Eq. (2.9) represent a vector $D(d_1, \ldots, d_n)$. In the ideal case, all the elements of this vector is 0. The Euclidian distance of a vector solution to the ideal case solution gives the satisfaction divergence calculated by Eq. (2.10). The satisfaction divergence is a measure of conflict intensity. More divergent solutions represent more intense conflicts, because the divergence is caused by conflicting decisions. However, the satisfaction divergence cannot be evaluated alone. It is evaluated with the global objective satisfaction. The system objective is to maximize the global objective satisfaction while minimizing players' satisfaction divergence. A solution with 0 divergence is not desirable if the global objective satisfaction is 0.

$$d_i = |s_k - s_{k'}| \,, \forall k, k' \in \mathbb{Z}^+, k' > k \qquad (2.9)$$

$$Divergence = \sqrt{\sum_{i=1}^{n} (d_i)^2} \qquad (2.10)$$

2.5 Monte Carlo Simulations

We performed a Monte Carlo simulation with the design problem of a multi-disc clutch system derived from the example studied in (Yannou et al., 2010). Three different player characters are defined with combinations of T_k and M_k design attitudes as shown in Table 2.1. A player with a higher T_k value compromises at a larger s_k level. A player that starts the design process with a higher M_k value intends to define more restrictive decision constraints. Thus, the restrictiveness of a player is higher if T_k and M_k values are larger. Each of the four cases is repeated 1000 times with randomly generated player characters sampled from Table 2.1. The same series of random seed numbers is utilized for each case, so the simulation results of four strategies are comparable. Also design agents are randomly processed in iterations, so the process sequence is completely independent from player characters. The precision value (*P*) is defined as 0.001. Thus, if the interval of any variable $X = [minX^t\ , maxX^t]$ does not contain $minX^t \times 1.001$, it is extracted from the loop at iteration *t*. Dynamic CSP is defined in C++ computer language and a CP solver library (IBM ILOG CP (2012)) is used to find solutions through its domain reduction and constraint propagation algorithms. The solve function of IBM ILOG CP is used to examine the feasibility of the model, so the D3 in Figure 2.3 is determined. If the solution of a propagated constraint returns 1, it means that the restricted model has at least one consistent solution and it is feasible. If it returns 0, it means that the model is unfeasible or the solution space is empty.

Table 2.1: Definitions of random characters

Restrictiveness	High	Moderate	Low
T_k:	(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1)	(0.45, 0.5, 0.55)	(0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4)
M_k:	(5, 6, 7, 8, 9)	(3, 4, 5, 6, 7)	(1, 2, 3, 4, 5)

2.5.1 Simulation Problem

The simulation problem is a design problem of a multi-disc clutch system that connects a weight lifter with an engine, followed by a gearbox (Figure 2.9). This is a complex design problem which contains 81 variables and 64 initial constraints. More details of this problem can be seen in Appendix A. Problem nomenclature and constant values are given in Table 2.2. There are four designers associated to the problem. Their sub-problems are given in Table 2.3. The piecewise constraints representing product preferences are shown in Table 2.4. Transitions are considered linear as in Figure 2.4. These define local objectives of the subsystems Player4 evaluates four design performance variable, the same weight is attributed to these performance variables: $s_4 = \sum_{i=1}^{4}(s_{4i} \times 0.25)$. The global objective is defined as: $Max(s_1 + s_2 + s_3 + s_4)$.

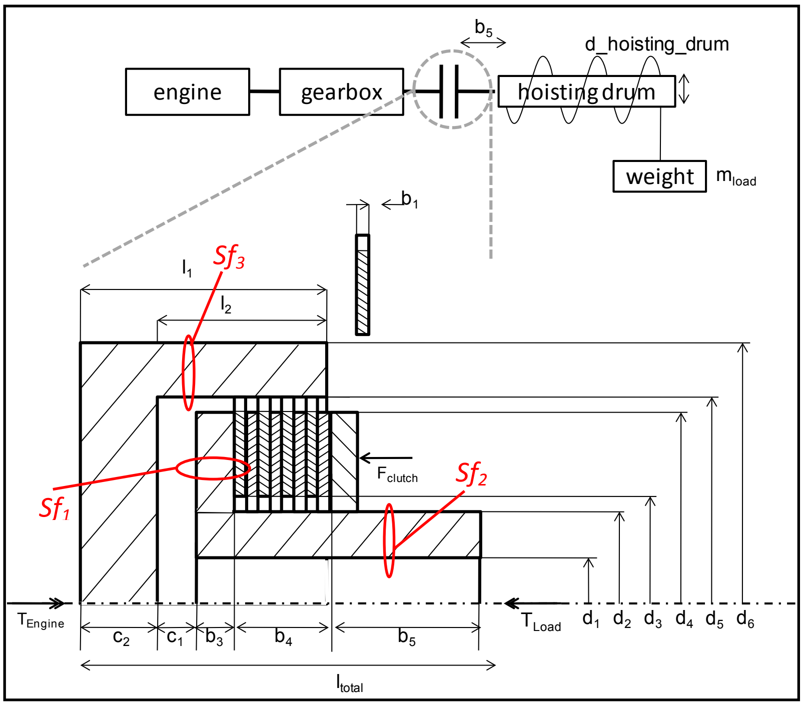


Figure 2.9: Multi-disc clutch system

Table 2.2: Clutch problem nomenclature & constants

c_1	Space between chassis and shaft, 10 mm
c_2	Thickness of chassis plate, 30 mm
b_1	Thickness
b_3	Width of shaft shoulder
b_5	Length of shaft, 300 mm
d_1	Diameter of bearing in shaft
d_2	Inner diameter of driven disc
d_3	Inner diameter of driving disc
d_4	Outer diameter of driven disc
d_5	Outer diameter of driving disc
d_6	Outer Diameter of chassis
d_m	Medium diameter of friction surface
p_1	Density of chassis and shaft material
p_2	Density of disc material
m_{load}	Weight of mass to be lifted, kg
m_s	Weight of whole system (clutch + engine), kg
m_e	Weight of engine, kg
f	Friction value between driving and driven discs
n	Number of revolutions of engine
t	Thermal conductivity of chassis and shaft material
i	Amount of friction pairs
T	Final temperature of the clutch, °C
σ	Stiffness of chassis and shaft material

P_m	Maximum allowed pressure on clutch discs
Sf_1	Safety against stress at position 1
Sf_2	Safety against stress at position 2
Sf_3	Safety against stress at position 3
Sf_p	Safety of discs material against pressure
s_k	Satisfaction of player *k*
s_{4i}	Satisfaction of Player4 from performance variable *i*
wb_k	Wellbeing of player *k*

Table 2.3: Clutch sub-problems

	Player1	**Player2**
Performance Variable Objectives	$Max(m_{load})$	$Min(m_s)$
Design Variable Objectives	$Max(m_{load})$	$Min(m_e, p_1, p_2, d_2, d_4, d_5, d_6, i, b_1, b_3)$ $Max(d_1, d_3)$

	Player3	**Player4**
Performance Variable Objectives	$Min(T)$	$Max(Sf_1, Sf_2, Sf_3, Sf_p)$
Design Variable Objectives	$Min(f, d_m, n, t, i)$ $Max(d_6, b_1, b_3, i)$	$Max(\sigma, P_m)$

Table 2.4: Clutch preferences

If $m_{load} \geq 3500$, $s_1 = 1$
If $m_{load} \leq 2000$, $s_1 = 0$
If $2000 < m_{load} < 3500$, $0 > s_1 > 1$
If $m_s \leq 150$, $s_2 = 1$
If $m_s \geq 500$, $s_2 = 0$
If $150 < m_s < 500$, $1 > s_2 > 0$
If ≤ 20 , $s_3 = 1$
If ≥ 150 , $s_3 = 0$
If $20 < T < 150$, $1 > s_3 > 0$
If $Sf_1 \geq 190$, $s_{41} = 1$
If $Sf_1 \leq 25$, $s_{41} = 0$
If $25 < Sf_1 < 190$, $0 > s_{41} > 1$
If $Sf_2 \geq 90$, $s_{42} = 1$
If $Sf_2 \leq 12$, $s_{42} = 0$
If $12 < Sf_2 < 90$, $0 > s_{42} > 1$
If $Sf_3 \geq 60$, $s_{43} = 1$
If $Sf_3 \leq 8$, $s_{43} = 0$
If $8 < Sf_3 < 60$, $0 > s_{43} > 1$
If $Sf_p \geq 10$, $s_{44} = 1$
If $Sf_p \leq 2$, $s_{44} = 0$
If $2 < Sf_p < 10$, $0 > s_{44} > 1$

2.5.2 Simulation Results

The average results of 1000 Monte Carlo simulations of each multi-disc clutch problem case are shown in Figure 2.10. In order to analyze the statistical significance of the results, we performed two tailed t-tests for each pair of cases. If the significance level is considered as 0.01, all the results are statistically significant ($p - value \cong 0$) except the iteration results of Case 2 and Case 4 ($p - value = 0.109$). The number of iterations and the number of failures are significantly larger when players define constraints on normalized design variables one by one (Case 1). This scenario results in the longest process time and the largest number of design conflicts.

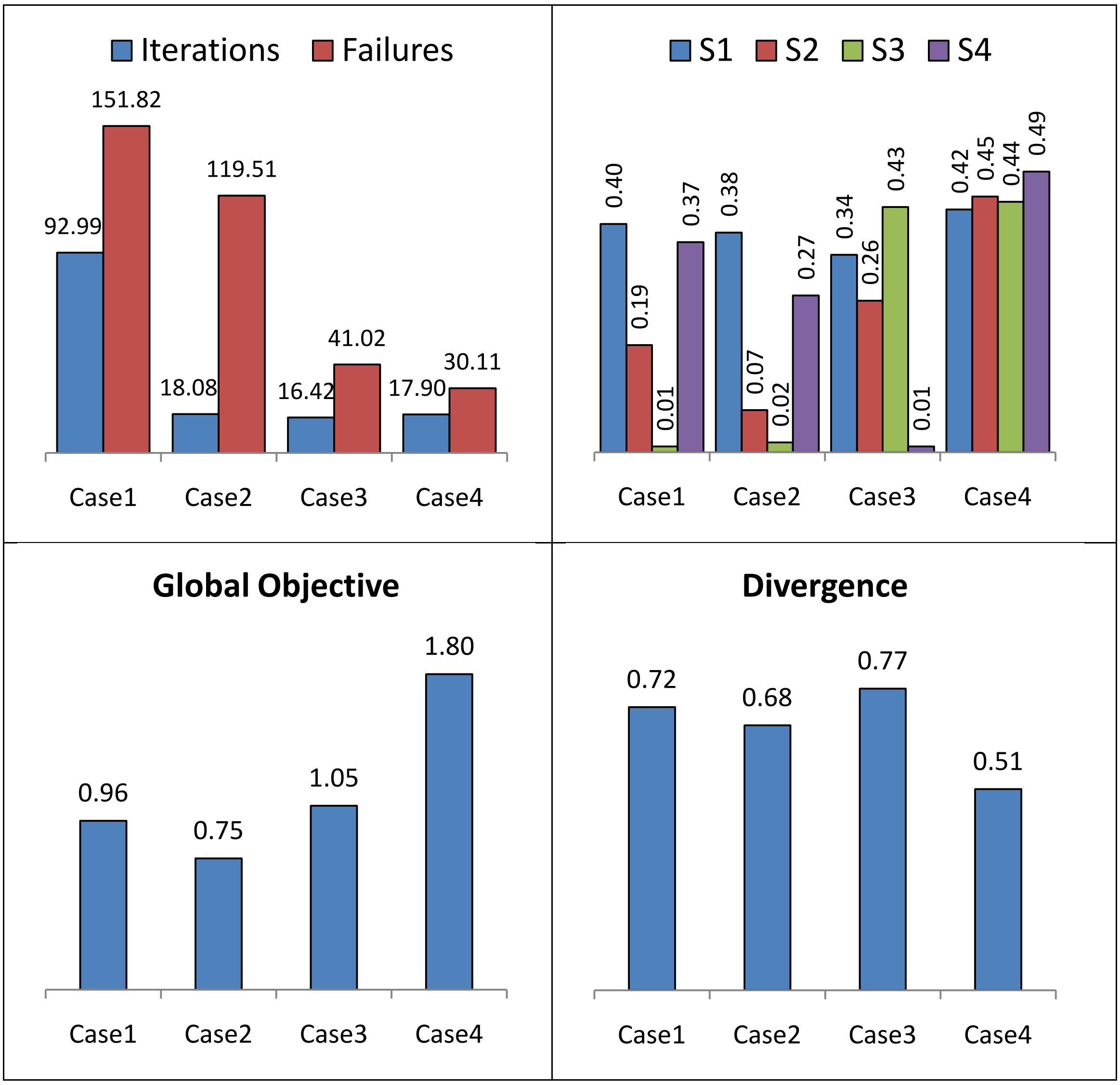


Figure 2.10: Average results of multi-disc clutch simulations

When normalized design variables are processed with the all-at-once approach (Case2), the process time and the number of conflicts decrease. However, Case 2 generates significantly more conflicts than Case 3 and Case 4. There are obvious satisfaction dominations on some players in Case1, Case2 and Case3. Here, the satisfaction domination is not character domination. It means that the design process has allowed a player to satisfy his/her local design objective significantly more, causing dissatisfaction to another player with a conflicting local objective. In Case 4, there is not an obvious domination, because players obtained closer average satisfaction values. Thus, Case 4 generates the least satisfaction divergence and the largest global satisfaction. Case 4 outperforms Case 1, Case 2 and Case 3 on all process performance criteria except Case 2 and Case 3 on the number of iterations.

2.6 Conclusions

In this paper, we proposed wellbeing indicators in a CSP processing that clearly bring significant advantages to concurrent designers when they take them into account for improving their local objective satisfactions. We performed CSP simulations of the wellbeing indicators in order to evaluate their contribution to the design process performance. The simulations are generated with Monte Carlo method where player attitudes and decision sequences are random. We compared the simulation results of a wellbeing controlled design (Case4) with three other cases: with Case1 where design actors modify only their local design variables one by one, with Case2 where design actors modify only their local design variables all-at-once and with Case3 where they modify only their performance variables.

In conventional design approaches, design actors perform a “blind design process” where they usually modify their local design variables with fuzzy intentions. They cannot evaluate precisely the contribution of their modifications to their performance variables, because the epistemic uncertainty is very high. Our simulation results show that this approach generates longer process time, more conflicts and satisfaction domination of one player on another player. With the CSP approach, a bottom-up design can be adopted where design actors can modify their performance variables directly. This is a simpler approach, because it avoids allocating design variables that are shared in coupled objectives of different design actors. However this approach results in an uncontrolled convergence of the solution space where individual satisfaction solutions are divergent and satisfaction domination of a player on another one is unavoidable.

When wellbeing indicators provide design information at any stage of the design process, and they are used to define design decisions, what is considered to be better or improved under epistemic uncertainty is precisely represented. Hence, design actors can improve their states in wellbeing equilibrium while reducing epistemic uncertainty with consistent decision constraints on the solution space. Consequently, design process performances are improved compared to other approaches: some design conflicts are prevented, the satisfaction domination is largely avoided and the intensity of conflicts is reduced. However, this approach can be applied on only measurable design systems where all the design aspects can be quantified. With this approach there is still some satisfaction divergence even if there is not any obvious satisfaction domination. This is because the design actor that obtains the best satisfaction and the design actor that obtains the worst satisfaction alternate with different combinations of attitudes of the design actors. Thus average satisfaction values are similar but the results of an individual case can be divergent. This means that the results are only influenced by designer attitudes and not by the design process. However, with modifying only design variables or performance variables the results are influenced by the design process itself, because even if the designer attitudes alternate there is obvious satisfaction domination.

In further works, more definitions will be provided on modeling design attitudes and simulating different characters of design actors; such as egoistic, altruistic. Our framework is capable of determining and preventing some design conflicts but it is not capable of resolving conflicts. Further, the process strategy will be improved to enable negotiating over constraints that are already accepted and compromising accepted constraints for resolving conflicts. This requires the detection of the source of conflicts that result in divergence.

2.7 References

Antonsson, E.K., Otto, K.N., 1995. Imprecision in Engineering Design. Journal of Mechanical Design 117, 25–32.

Balling, R.J., Sobieszczanski-Sobieski, J., 1996. Optimization of coupled systems - A critical overview of approaches. AIAA Journal 34, 6–17.

Brailsford, S.C., Potts, C.N., Smith, B.M., 1999. Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research 119, 557–581.

Canbaz, B., Yannou, B., Yvars, P.-A., 2011. A New Framework for Collaborative Set-Based Design: Application to the Design Problem of a Hollow Cylindrical Cantilever Beam, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, Washington, D.C., pp. 197–206.

Canbaz, B., Yannou, B., Yvars, P.-A., 2012. Constraint Programming Simulation of a Distributed Set- Based Design Framework with Control Indicators, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, Chicago, IL.

Chanron, V., Lewis, K., 2005. A study of convergence in decentralized design processes. Res Eng Design 16, 133–145.

Cramer, E., Dennis, J., Frank, P., Lewis, R., Shubin, G., 1994. Problem Formulation for Multidisciplinary Optimization. SIAM Journal on Optimization 4, 754–776.

Dechter, R., Dechter, A., 1988. Belief Maintenance in Dynamic Constraint Networks, in: American Association for Artificial Intelligence. Presented at the 6th National Conference on Artificial Intelligence (AAAI-88), pp. 37–42.

Faltings, B., 1994. Arc-consistency for continuous variables. Artificial Intelligence 65, 363–376.

Fathianathan, M., Panchal, J.H., 2009. Modelling an ongoing design process utilizing top-down and bottom-up design strategies. Proceedings of the Institution of

Mechanical Engineers, Part B: Journal of Engineering Manufacture 223, 547–560.

Granvilliers, L., 2012. Adaptive Bisection of Numerical CSPs, in: Milano, M. (Ed.), Principles and Practice of Constraint Programming, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 290–298.

IBM, 2012. IBM ILOG CPLEX CP Optimizer for Constraint Programs - Features and benefits [WWW Document]. URL http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/about/

Kim, H.M., Michelena, N.F., Papalambros, P.Y., Jiang, T., 2003. Target Cascading in Optimal System Design. Journal of Mechanical Design 125, 474.

Lewis, K., Mistree, F., 1998. Collaborative, Sequential, and Isolated Decisions in Design. Journal of Mechanical Design 120, 643.

Lottaz C, S.R. and S.I., 2000. Increasing Understanding During Collaboration Through Advanced Representations [WWW Document]. URL http://www.itcon.org/cgi-bin/works/Show?2000_1

Lottaz, C., Smith, I.F.., Robert-Nicoud, Y., Faltings, B.., 2000. Constraint-based support for negotiation in collaborative design. Artificial Intelligence in Engineering 14, 261–280.

Mackworth, A.K., 1977. Consistency in networks of relations. Artificial Intelligence 8, 99–118.

Malak, R.J., Aughenbaugh, J.M., Paredis, C.J.J., 2009. Multi-attribute utility analysis in set-based conceptual design. Computer-Aided Design 41, 214–227.

McKenney, T.A., Kemink, L.F., Singer, D.J., 2011. Adapting to Changes in Design Requirements Using Set-Based Design. Naval Engineers Journal 123, 66–77.

Messac, A., 1996. Physical programming - Effective optimization for computational design. AIAA Journal 34, 149–158.

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

Panchal, J.H., Fernández, M.G., Paredis, C.J.J., Allen, J.K., Mistree, F., 2007. An Interval-based Constraint Satisfaction (IBCS) Method for Decentralized, Collaborative Multifunctional Design. Concurrent Engineering 15, 309–323.

Papalambros, P.Y., Michelena, N.F., Kikuchi, N., 1997. Distributed Cooperative Systems Design, in: Proceedings of the 11th International Conference on Engineering Design. Tampere, Finland, pp. 265–270.

Park, H., Michelena, N., Kulkarni, D., Papalambros, P.Y., 2001. Convergence Criteria for Hierarchical Overlapping Coordination of Linearly Constrained Convex Design Problems. Computational Optimization and Applications 18, 273–293.

Parsons, M.G., Singer, D.J., Sauter, J.A., 1999. A hybrid agent approach for set-based conceptual ship design, in: Proceedings of 10th International Conference on Computer Applications in Shipbuilding. Cambridge, MA.

Sam-Haroud, D., Faltings, B., 1996. Consistency techniques for continuous constraints. Constraints 1, 85–118.

Sobek, D.K., Ward, A.C., Liker, J., 1999. Toyota's Principles of Set-Based Concurrent Engineering. Sloan Management Review 40, 67–83.

Sobieszczanski-Sobieski, J., Barthelemy, J.F.M., Giles, G.L., 1984. Aerospace engineering design by systematic decomposition and multilevel optimization, in: 14th Congr. of the International Council of the Aeronautical Sciences (ICAS). Toulouse, France.

Sobieszczanski-Sobieski, J., Haftka, R.T., 1997. Multidisciplinary aerospace design optimization: survey of recent developments. Structural Optimization 14, 1–23.

Wang, J., Terpenny, J., 2003. Interactive evolutionary solution synthesis in fuzzy set-based preliminary engineering design. Journal of Intelligent Manufacturing 14, 153–167.

Ward, A.C., Liker, J., Sobek, D.K., Cristiano, J.J., 1994. Set-based concurrent engineering and Toyota, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 79–90.

Wood, W., 2001. A view of design theory and methodology from the standpoint of design freedom, in: Proceedings of the ASME Design Engineering Technical Conference. ASME, pp. 345–355.

Yannou, B., Harmel, G., 2004. A Comparative Study of Constraint Programming Techniques Over Intervals in Preliminary Design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 189–198.

Yannou, B., Mazur, C., Yvars, P.-A., 2010. Parameterization and dimensioning of the multi-disc clutch in the CO4 environment, Technical report (Cahier d'Etudes et de Recherches), Ecole Centrale Paris, September 7th 2010, available at "http://www.lgi.ecp.fr/uploads/Recherche/CO4Multi_DiscClutch_Technical_D ocument" on July 9th 2013.

Yannou, B., Yvars, P.-A., Hoyle, C., Chen, W., 2013. Set-based design by simulation of usage scenario coverage. Journal of Engineering Design. DOI:10.1080/09544828.2013.780201.

Yvars, P.-A., 2009. A CSP approach for the network of product lifecycle constraints consistency in a collaborative design context. Engineering Applications of Artificial Intelligence 22, 961–970.

Zhao, L., Jin, Y., 2003. Work Structure Based Collaborative Engineering Design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 865–874.

Chapter 3: Preventing Design Conflicts in Distributed Design Systems Composed of Heterogeneous Agents

Baris Canbaz, Bernard Yannou, Pierre-Alain Yvars

This paper has been submitted to *Engineering Applications of Artificial Intelligence.*

Abstract

In distributed design systems, while designers are connected to each other through dimensioning couplings, they have limited control over design and performance variables. Any inconsistency among design objectives and working procedures of heterogeneous designers interacting in the design system can result in design conflicts due to these couplings. Modeling design attitudes can help to understand inconsistencies and manage conflicts in design processes. We extend the conventional bottom-up or design supervision approach through agent-based attitude modeling techniques to a more powerful level. In our model, design agents can set requirements directly on their wellbeing values that represent how their design targets are likely to be met at a given moment of the design process. Some design conflicts can in this manner be prevented at an earlier phase of the design process. Set-based design and constraint programming techniques are used to explore the overall performance of stochastic design collaborations on a product modeled with uncertainties at a given moment of the design process. Monte Carlo simulations are performed to evaluate the performance of our set-based thinking approach, providing a variety of agent attitudes. The results show that the number of design conflicts occurring during the design process and the intensity of design conflicts are both reduced through our collaborative design platform.

3.1 Introduction

Design processes of complex products currently involve considerable effort and expertise from different disciplines. Multiple designers from different disciplines are thus involved in performing collaborative design. The design model converges to a solution through a series of collaborative activities performed during the design process. Since the design problem has multidisciplinary boundaries, a distributed design approach can be adopted. In distributed design systems, the system is decentralized; the global problem is decomposed into sub-problems and distributed to subsystems consisting of one or several designers (Papalambros et al., 1997). Subsystems have limited control over the design variables because of their limited expertise and responsibility. In a sub-problem there are three main problem elements: design variables that can be controlled, design performances that are evaluated and constraints that must be respected. The rest of the global problem excluding a specific sub-problem does not concern the specific sub-system, but it can be only observed if it is shared and necessary. Distributed design tasks allocated to sub-problems are executed concurrently by subsystems, the global problem converging to a global solution (Zheng et al., 2011).

In the ideal case, true concurrency is expected from distributed design systems where designers can perform their design activities independently. In reality, designers are related to each other through couplings between their sub-problems. Couplings can result in conflicts among designers if some inconsistencies are presented in the design system. Inconsistencies arise from design attitudes reflected by subsystems during the design process. The most significant inconsistency occurs between design objectives of subsystems. Typically, a design problem contains multiple conflicting objectives, so subsystems are forced to make trade-offs. Working procedures of designers influence the performances of others, and inconsistencies present in these working procedures can negatively impact the global solution (Zhao and Jin, 2003). For instance, a designer restricting the design model more rapidly or earlier than others could influence the model more. Subsequent designers are forced to deal with a restricted model which cannot satisfy their own design objectives. If the number of conflicts and intensity of the conflicts increase; the performance of the design process decreases, because individual design objectives are not satisfied in equilibrium. Some significant attempts have been made to coordinate and resolve existing conflicts in distributed systems. Zheng et al. (2011) propose to resolve conflicts by integrating resultant models of conflicting Boolean decisions in individual sub-problems of distributed computer-aided design. Kwon and Lee (2002) define a multi-agent based model that integrates a coordination mechanism. This can manage conflicting agents in a decentralized enterprise in order to resolve interdepartmental conflicts. Koulinitch and Sheremetov

(1998) define a constraint-based dynamic design system model that includes facilitator agents which are responsible for coordination and conflict resolution during the design process. When a conflict occurs amongst design agents, facilitator agents send messages to relax some constraints until a consistent solution is obtained. Huang et al. (2006) develop a fuzzy interactive multi-objective optimization model for engineering design. The collaborative relationships among the objectives are improved with adjusting the threshold of satisfaction degree and weighting coefficients of objectives. The least conflicting solution is therefore selected among the generated set of Pareto optimal results. The selected solution gives the maximum satisfaction degree and the minimum divergence of the individual satisfactions of local objectives. Yvars (2009) proposes a collaborative design system where decisions of distributed designer agents are represented with constraints added to the model dynamically. Constraints restricting the design model restrict also the degree of freedom of agents, so that they cannot add anymore constraints to the design model. This results in conflicts that are represented as unfeasible models. Design conflicts are resolved by detecting a compromise solution that maximizes the number of accepted constraints by removing some constraints from the model. While these approaches focus on resolving conflicts that have already occurred, they overlook the idea of preventing and avoiding potential conflicts that have not yet occurred in the process. They interrogate the issue at a late phase of the problem, because the avoidance of a conflicting problem is usually more efficient and less time-consuming than the resolution of a conflicting problem. The approaches outlined above also fail to take into account attitude models of heterogeneous agents. Modeling design attitudes can help understand the design inconsistencies resulting in design conflicts, and as a result certain collaboration strategies can be defined with attitude models.

The technique chosen for modeling the design process significantly affects the collaborative solution emerging from different sub-problems. Devendorf and Lewis (2011) show that the stability of a distributed design system depends on how the process architecture is formed. Two main approaches can be adopted for global design process modeling. These are the top-down design approach and the bottom-up design approach (Fathianathan and Panchal, 2009). In the top-down design approach, decisions are made for parameterization of design variables in order to find detailed solutions that satisfy designer objectives. This approach is considered as a transition from an abstract level to a detailed level: in complex design problems, the effect of any parameter on the solution is usually abstract until the parameter is tested and a detailed solution is obtained. In contrast, the bottom-up design approach consists in defining detailed solutions to identify values of the design variables. With this bottom-up design approach, designers can make decisions on their design performances. The top-down design approach requires detailed decomposition of the problem where all the relations between variables are explicit. However, this may not be possible when the complexity of the design problem is very high and the problem contains too many couplings. Therefore, the effect of the decisions about design variables on design performances is highly uncertain, especially in early design phases. Engineering project failures can increase when it is not possible to predict the effect of the modifications because of the presence of intense couplings in complex design problems. Chanron and Lewis (2005) highlight the difficulty of allocating design variables to subsystems in a coupled problem where the same design variables influence the design performances of several subsystems. The allocation technique is critical, because it can influence the design quality (Kim et al., 2003) or the design process performance (Park et al., 2001). Fathianathan and Panchal (2009) propose the adoption of a bottom-up design approach when these limitations arise from a top-down design approach.

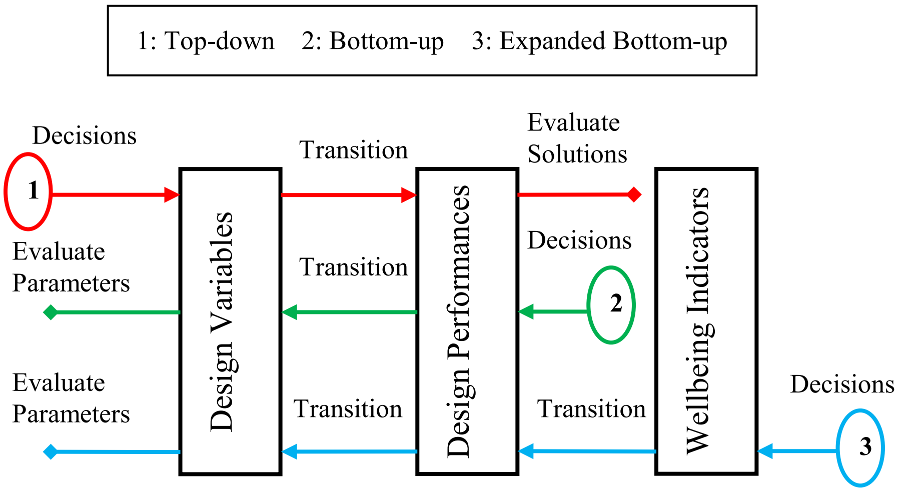


Figure 3.1: Comparison of process approaches

In this paper we extend the bottom-up design approach with agent-based attitude modeling techniques. A wellbeing indicator is presented that shows how the preference objectives of various designers are satisfied. Figure 3.1 shows the comparison of our extended bottom-up design approach with the traditional bottom-up approach and the top-down approach. In the top-down design approach, alternatives are generated first by making decisions on the design variables, and emerging solutions are subsequently evaluated considering design performances. In the bottom-up approach, solutions are generated by making decisions on design performance values and the parameters emerging from these values are evaluated to see if they are feasible or if they violate the problem constraints. Thus, trade-offs are made on design performance values. Traditionally, the bottom-up design approach is modeled at the design problem level: it starts at the lowest level of the physical problem. However it does not include modeling the preferences of designers emerging from their design attitudes. We think that modeling design attitudes and including them at the bottom of the design approach will enable better control of collaborative convergence, because trade-offs can be made directly on satisfaction values of designer preferences. Therefore, the design conflicts can be reduced.

A common design issue, regardless of the design process approach used, is the presence of epistemic uncertainty due to the imprecision caused by the lack of knowledge about the final decision (Parry, 1996). According to Malak et al. (2009) this issue requires representing the uncertainty with imprecise intervals/sets and delaying uncertain decisions to later process stages when the information about the related decision becomes available. In this paper, we use the set-based design (SBD) concept to simulate the process performance of the extended bottom-up design approach modeled with agent attitudes. In Section 3.2 we discuss the ability of SBD and constraint satisfaction problem (CSP) techniques to manage imprecision in design. In Section 3.3 the attitudes of design agents in multi-agent systems and egoistic and altruistic agent characters emerging from dynamic attitudes are considered. Our agent-based SBD model is introduced in Section 3.4 and the CSP simulation process of this model is presented in Section 3.5. Monte Carlo simulations of our approach are performed on a design problem which involves variable agent characters composed of variable design attitudes that define how design agents react during the design process. The sequence of the agent reactions is stochastic. Problem definitions and simulation results are presented in Section 3.6.

3.2 SBD and CSP Techniques

In coupled and conflicting design problems, especially in preliminary design processes, variables cannot be crisply defined due to the lack of information about the decision consequences (Antonsson and Otto, 1995; Yannou, 2004). Even so, in deterministic design methods, crisp values are attributed to variables, so trade-offs are made on design point solutions. Hence, deterministic methods simplify and restrict the design problem in order to optimize it. However, this requires making radical decisions before the information about the decision becomes certain. Therefore, important uncertainty aspects are overlooked. Alternatively, SBD concept considers the design process as an ongoing evolution of non-crisp concurrent decisions (Sobek et al., 1999; Ward et al., 1994). Variables are represented with imprecise values in domains (intervals for real variables), so epistemic uncertainty can be propagated and evaluated. This concept allows information to be gathered before making decisions on the design model, and decisions to be delayed when the information is not certain. The delayed decisions are reconsidered at later process stages where more information has been gathered due the reduction of epistemic uncertainty through earlier decisions. This approach provides flexibility of modifications and higher adaptability to changes as shown by Wang and Terpenny (2003), as well as robustness to design errors as shown by Parsons et al. (1999). Process time is consequently reduced due to a decrease of repetitive design activities and loopbacks.

If SBD has been principally adopted at a managerial level, it is only recently that this concept has been adapted using CSP definitions at a technical solution level e.g. (Meyer and Yvars, 2012; Panchal et al., 2007; Yannou and Harmel, 2006). A CSP is defined with sets of variables, sets of domains that contain the allowable values of variables and sets of constraints that restrict the problem (Montanari, 1974). The Cartesian product of the variable intervals defines a multidimensional space that contains the consistent values which respect the constraints. A decomposed design problem can be defined with three spaces: the design space defined by design variables, the performance space defined by design performance variables, and the solution space that contains both design and performance spaces. Design decisions are represented with constraints restricting the solutions space, so the epistemic uncertainty is reduced and the remaining solution space is precisely determined with domain reduction/filtering algorithms of constraint programming (CP) techniques. Yannou and

Harmel (2004) show how CP techniques can compete with and outperform probabilistic and fuzzy methods on managing imprecision in design. CP techniques allow the bottom-up design approach with enabling constraint definitions on value occurrences. For instance X and Y are integer variables with domains $D(X) = [15\,,25]$ and $D(Y) = [10\,,20]$ and $Z = X \times Y$ is a value occurrence. If a constraint is defined on Z, its consistency is evaluated and the inconsistent values of X and Y are extracted from their domains. If $Z \leq 200$ the domains of the variables are reduced to $D(X) = [15\,,20]$ and $D(Y) = [10\,,13]$. Current CSP definitions are able to support a bottom-up SBD, but we are not aware of any CSP platform that includes collaboration indicators derived from design attitudes.

3.3 Attitudes of Design Agents in MAS

Through agent-based modeling, many complex phenomena can be considered as systems of autonomous agents that follow simple rules of repetitive, cooperative and competitive interactions. Thus multi-agent system (MAS) simulation is considered as an appropriate approach to investigate complex emergent systems. For instance, MASs have been used for social simulation (Caballero et al., 2011), for modeling bounded rational agents (Lin et al., 2008), and for organization of societies (Rodriguez et al., 2011). Agents are sub-systems that are situated in an environment, and in order to satisfy their design objectives they perform autonomous actions (Wooldridge and Jennings, 1995). In the environment they are social, so they can communicate and interact; they are reactive, so they can perceive the environment and respond to the change in the environment; and they are pro-active, so they can take initiatives by their goal-directed attitudes. In MAS, agents can reflect different attitudes that represent the reactions of agents to uncertainties of complex dynamic domains (Goyal, 2005). The widely deployed architecture of an agent, the belief-desire-intention (BDI) paradigm, is developed by Bratman et al. (1988). BDI views the system as it is emerging from agents with different mental attitudes. The emergent mental attitudes construct the system behavior and are important for the optimal performance of the system. Beliefs correspond to the information emerging from the analysis of the model. Desires correspond to the objectives of the agent and the tasks allocated to it. In a complex emergent system, agents are not able to satisfy all their desires at the same time, so they are forced to make trade-offs and compromise. Intentions correspond to the choices of the agent for some desires when compromise is necessary. Actions of choosing desires are intentions: an agent makes intentions until the desire is satisfied or until the agent believes that the intention is no longer feasible (Cohen and Levesque, 1990). Agents perceive their environment through sensors and act upon that environment through effectors. The system between perception and action consists of their attitudes. An agent is stimulated by the analysis of the model and through its belief, desire and intention architecture its attitudes are defined, so the agent performs actions (Figure 3.2). Finally the new form of the model is synthesized following the actions.

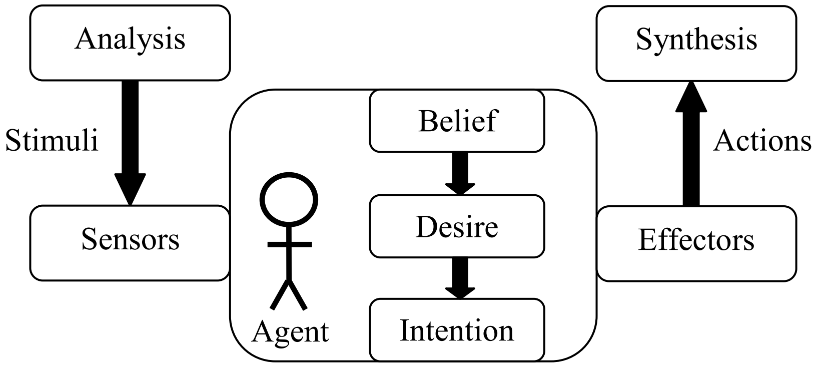


Figure 3.2: BDI paradigm

A distributed design system is an emergent system and it can be simulated as an MAS. In a distributed design system, the stimuli are sent to agents by the dynamic design model and agents react through defining decision constraints in the dynamic design model. Design attitudes are bounded and design agents need to interact and collaborate. Attitudes of design agents determine when and how their decision constraints are defined during the progress of the design process. This shapes the decision making process and collaborative convergence. The most widely employed decision making model in MAS is the multiple attribute utility theory (MAUT) which evaluates multiple performances. The decision maker agent attempts to maximize the utility function which aggregates all the performances. The utility is used to evaluate solutions while the analysis of trade-offs between alternatives is represented as weighted formulae. Decision makers can also rank alternatives and define preferences on one alternative over another. Preferences reflect agents' objectives and can be prioritized with constraints. Thus, constraints are used to make decisions either statically or dynamically. Therefore, a joint solution is generated by modifying the design model iteratively.

The design system is composed of different people each with different characters. The character of an agent is the combination of its attitudes, and it can be used to establish strategies in order to achieve optimal interactions between various agents (Castelfranchi et al., 1998). However, the attitudes of different agents can result in conflicting activities. This problem usually requires coordination and cooperation of agents' attitudes. MAS can simulate the coordination of different agents composed of different attitudes. The emergent behavior of the system consists of different compositions of altruistic and egoistic behaviors of every agent in the system (Pita and Lima Neto, 2007). Egoistic behaviors are actions that are motivated by self-interested gains, while altruistic behaviors are motivated by the gain of others, such as the pleasure obtained from others' pleasure. Altruism can also be viewed as sacrificing one's own good for the benefit of the group that one belongs to. While egoistic actions can cause harm to the other agents, altruistic actions help the others. A mutual defection may be the rational solution of the agents, but it is neither the most beneficial one for the global benefit nor even for individual benefits. Bazzan et al. (2002) simulate the effects of altruism among agents playing the Iterated Prisoner's Dilemma. They conclude that egoistic agents maximize their benefits only in the short term, but they compromise their performances in the long term. Xianjia and Weibing (2009) propose a method to investigate the evolutionary outcome of the behaviors of players with egoistic or altruistic preference in an iterated prisoner's dilemma. Their results show that egoism can cause defection, and altruism can increase the performance of cooperation. Jennings and Campos (Jennings and Campos, 1997) conclude that the overall performance of the system can be increased if agents are sometimes allowed to work for the benefit of others. Since agents are autonomous and have different knowledge and resources, cooperation attitudes are conditional to the environment and are dynamic through the allocation of time and resources. Agents are therefore heterogeneous, and it is almost impossible to define optimal agent attitudes. To maintain cooperation among heterogeneous agents, social norms and collaborative strategies should be adopted upstream in the system.

3.4 Agent-based SBD Model

We define the extended bottom-up design approach as an agent-based SBD model that considers the design attitudes of interacting agents. We first define the design process before presenting the design attitudes and the control indicators that derive from these attitudes.

3.4.1 Design Process of Agents

In the preliminary design phase, the solution space is very large. While the solution that designers find at the end of the design process is presented in the initial solution space, this solution is not known at the initial state. This implies a very high epistemic uncertainty. CSP definitions can be used to model designer actions. Designer actions are considered as decision constraints defined on the solution space iteratively. The design model is therefore dynamic, evolving with the actions representing decisions. Hence, a collaborative point solution emerges from the converging solution space while the epistemic uncertainty is reduced iteratively during the design process stages. We model an agent-based design process in order to understand both how the epistemic uncertainty is reduced, and how the solution space converges collaboratively. The design process model is shown in Figure 3.3.

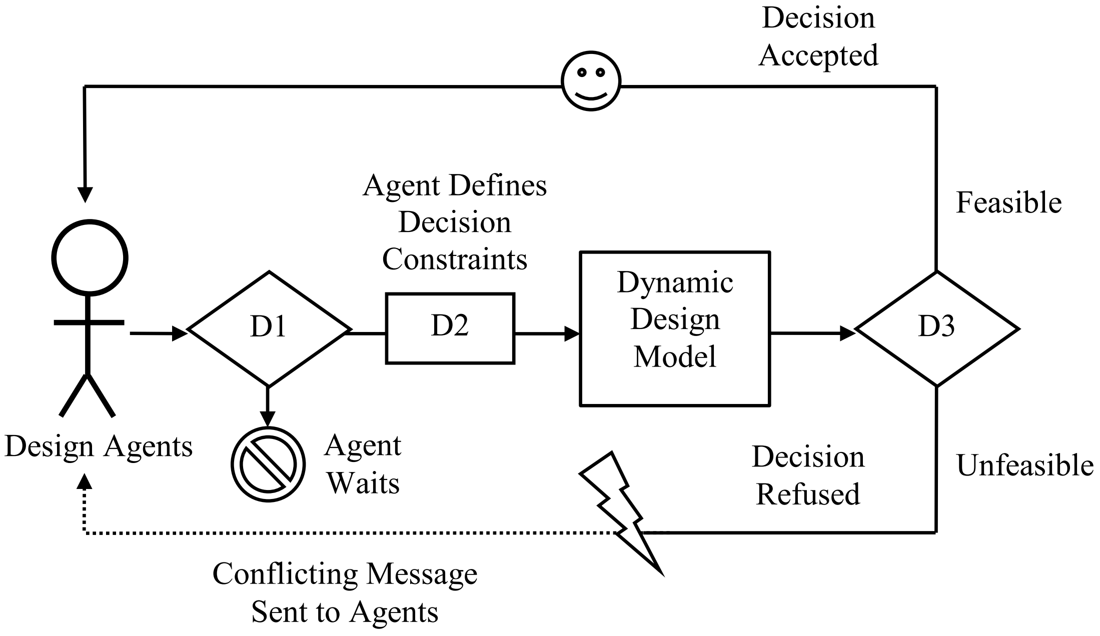


Figure 3.3: Design process of agents

Design agents make three decisions during the process: these are shown as D1, D2, and D3. D1 and D3 are Boolean decisions and D2 is a “how” decision.

D1: Define a decision constraint or not.

D2: How the decision constraint is defined.

D3: Accept the decision constraint or not.

During a process stage, agents evaluate the solution space to decide whether they will define a decision constraint, or wait. If they decide to define a decision constraint, next they decide how the constraint is defined. When the constraint is defined in the dynamic design model, the model’s feasibility is tested. After the definition of the constraint, if there is at least one solution remaining in the solution space, the constraint is consistent for the model and it is accepted collaboratively. If the solution space is empty, then it is refused and rejected. The consistency of the constraint depends on the previously accepted constraints that have been defined by the process stage agent and other agents. In addition, the consistency of the constraint depends on how it is restrictively defined, and the nature of the initial problem. If the constraint is very restrictive, it is probably refused whether there is a previously accepted constraint or not. Therefore, D3 is a collaborative decision which has emerged from the collaborative behavior of the design system. In contrast, D1 and D2 are individual decisions defined by the individual design attitudes of the agents. When a constraint is refused, it is considered as a potential conflict because the agent’s desires may not be sufficiently satisfied. The degree of the conflict can be evaluated by the divergence of the agents’ individual objective satisfaction values. The solution space is shared and design objectives are typically conflicting. If an agent can satisfy its desires, it results in dissatisfaction of another agent with conflicting objectives. When their satisfaction solutions diverge - for instance the solution of an agent with a very low satisfaction and the solution of another agent with a very high satisfaction - the conflicts increase in intensity. The conflict is reasonable if only the agent’s desires are not sufficiently satisfied. Our proposition is to evaluate design attitudes with a BDI model and evaluate agents’ states with control indicators called wellbeing indicators. Wellbeing indicators are derived from the desires of the agents reflected on the beliefs of the agents. They enable a bottom-up design process where convergence is controlled, with defining decision constraints impacting directly on the wellbeing intervals instead of on design variable intervals.

3.4.2 Attitudes of Agents and Control Indicators

Design space emerges from the intervals of design variables modified dynamically during the design process. This represents the dynamic design model. Analysis of the dynamic design model stimulates design agents and triggers their BDI mechanism. Figure 3.4 shows design agents' BDI mechanism.

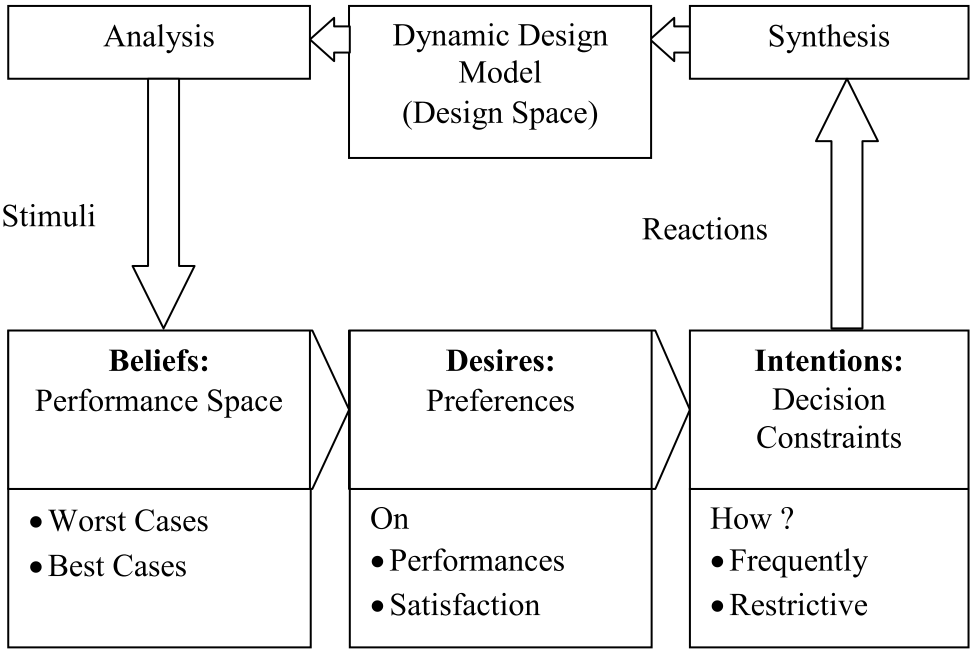


Figure 3.4: BDI mechanism of design agents

Beliefs of design agents are reflected with the intervals of the design performances emerging from the dynamic design model. The bounds of the intervals of the design performances represent the worst possible cases and the best possible cases for the corresponding design performances. Since actions of the design agents are bound through couplings, the intervals propagate some uncertainty. The worst possible cases and the best possible cases depend on the actions of the other agents. Therefore design agents define their desires to adopt the performance values. Desires are design agents' preferences on two factors: design performance alternatives and the satisfaction obtained by design performances. While preferences on design performances reflect agents' attitudes for satisfaction obtained from the alternatives, preferences on satisfaction represent agents' attitudes for compromise. Beliefs and preferences of design agents lead design agents to define their intentions in order to reduce the solution space by improving their worst cases. Intentions are reflected with how frequently and how restrictively their decision constraints are defined. Design agents react to the emerging performance space through defining decision constraints into solution space. These modifications synthesize the next design space in the dynamic process.

We define an agent k, A_k, as an entity with four different attitudes: $A_k\langle Pr_k, T_k, F_k, M_k\rangle$. Pr_k is the set of preferences of the agent on performance values. T_k is the compromise threshold value of the agent, representing the preference of the agent on the satisfaction values for compromise. F_k is the average frequency of the agent for defining constraints in the model and M_k is the coefficient of restriction of the constraints defined by the agent, reflecting the restrictiveness of the decision constraints defined by the agent.

3.4.2.1 Preferences and Satisfaction

Preferences of an agent about design performances can be modeled as a satisfaction function. The list of preferences of an agent A_k on its performances creates the Pr_k attitude. Complete dissatisfaction is represented by 0 on the scale, while complete satisfaction is represented by 1. Design agents are moderately satisfied in the transition between fully satisfied and fully dissatisfied states. In this paper, we assume that the transition is linear; however nonlinear satisfaction functions can be adopted for different studies. We integrate piecewise constraints into the model in order to define information about performance preferences without restricting the solution space. For example, one objective of an agent k could be minimizing a performance i; the agent is fully satisfied by a performance value below or equal to P_1 and fully dissatisfied by a performance value above or equal to P_2. It is assumed that there is a linear transition between these two preference values. s_{ki} is the satisfaction value of the agent k obtained by the performance i and v_i is the performance value of the performance i. The corresponding integrated piecewise constraints are:

If $v_i \leq P_{1i}$, $s_{ki} = 1$ (3.1)

If $v_i \geq P_2$, $s_{ki} = 0$ (3.2)

If $P_1 < v_i < P_2$, $1 > s_{ki} > 0$ (3.3)

In the SBD framework, all the variables are defined with intervals instead of points. The design process progresses with time and the intervals are reduced through the decision constraints defined on the solution space during the progress. Thus the design process is composed of design stages where agents take actions. At process stage t, performance i has a minimum value x_i^t and a maximum value y_i^t, the interval of the performance i at process stage t being $[x_i^t , y_i^t]$. Since the performance is defined with an interval, we obtain an interval for the satisfaction of agent k from the performance i at stage t: $s_{ki} = [mins_{ki}^t , maxs_{ki}^t]$ where $mins_{ki}^t$ is the minimum satisfaction and $maxs_{ki}^t$ is the maximum satisfaction obtained within the interval $[x_i^t , y_i^t]$. This phenomenon is illustrated in Figure 3.5. Piecewise constraints are the same as above. Minimum satisfaction is obtained at point A and maximum satisfaction is obtained at point B. During the process while uncertainty is reduced, agents can observe the potential maximum and minimum satisfaction values from performances. When design agents have several design performances to evaluate, they can assign weights to their satisfaction values considering the importance of the performances for their job. As individual performance satisfaction values are aggregated, general satisfaction states of agents can be observed.

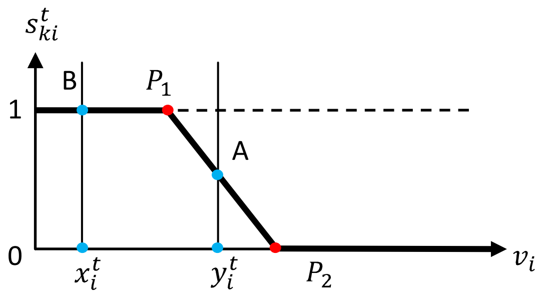


Figure 3.5: Intervals of satisfaction function

3.4.2.2 Compromise Threshold and Wellbeing

In a coupled design system, it is highly unlikely that all the design agents will be fully satisfied. Since design objectives are conflicting, a decision constraint defined to increase the minimum satisfaction value of an agent will decrease the maximum satisfaction value of another agent. Thus, the convergence of satisfaction intervals is bilateral, and design agents are forced to compromise at a certain level on their satisfaction values where maximum and minimum satisfactions are as close as possible. Figure 3.6 shows a clear example of this phenomenon where Agent 1 and Agent 2 have conflicting objectives, such as decreasing the mass and increasing the volume of a product. Agent 1 defines C_1 and C_3 and Agent 2 defines C_2 and C_4 at different process stages, in order to improve their satisfaction states. However, these constraints decrease the other agent's maximum satisfaction value.

Since agents cannot be aware of the other agents' actions, the convergence propagates some uncertainty. Thus, design agents can reflect an attitude of desiring a value in which they may compromise. While preferences on design performances reflect the desires of agents on product specifications, preferences on satisfaction values obtained by these performances reflect the desires about process convergence. The preference about the satisfaction value is called compromise threshold T_k, and it defines the compromise attitude of an agent. This compromise threshold value represents the satisfaction value that an agent wants to guarantee. The agent wants the solution to converge at least to this value.

The agent defines decision constraints considering the preferences Pr_k in order to increase the satisfaction obtained from the model until the minimum satisfaction value reaches the agent's satisfaction preference. This introduces a condition for making decisions during the design process. If the minimum satisfaction of an agent by the model reaches or exceeds value T_k , then the agent passes to the compromise state. In the compromise state, agents stop adding decision constraints to the model, so this leaves space to the other agents.

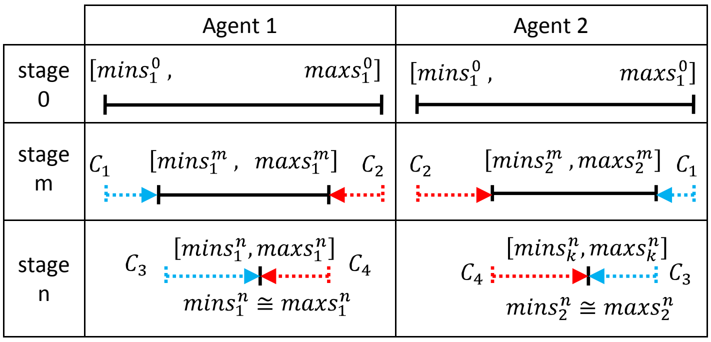

Agent 1
Agent 2
stage 0
$[mins_1^0 ,$ $maxs_1^0]$
$[mins_1^0 ,$ $maxs_1^0]$
stage m
C_1 $[mins_1^m ,$ $maxs_1^m]$ C_2
C_2 $[mins_2^m , maxs_2^m]$ C_1
stage n
C_3 $[mins_1^n, maxs_1^n]$ C_4
$mins_1^n \cong maxs_1^n$
C_4 $[mins_k^n, maxs_k^n]$ C_3
$mins_2^n \cong maxs_2^n$

Figure 3.6: Bilateral convergence

Satisfaction values are normalized through dividing them by the compromise threshold value; this provides wellbeing states Eq. (3.4, 3.5). Wellbeing states represent global states of design agents; they show if an agent suffers from not being able to approach the compromise state or if an agent could have performed modifications to the model and thus approached the compromise state. Wellbeing is defined with an interval $wb_k = [minwb_k^t , maxwb_k^t]$ where the minimum value is the minimum wellbeing indicator and the maximum value is the maximum wellbeing indicator. The wellbeing interval converges through the progress of the design process. If the minimum wellbeing value is larger than or equal to 1, then the agent is in a perfect wellbeing state. The worst wellbeing state is when the value is equal to 0.

$$minwb_k^t = \frac{mins_k^t}{T_k} \qquad (3.4)$$

$$maxwb_k^t = \frac{maxs_k^t}{T_k} \qquad (3.5)$$

3.4.2.3 Frequency

Design agents define decision constraints at an average frequency F_k per process stage. This attitude, dependant on agent character, reflects if agents intend to restrict the solution space more frequently or less. Ph_k is the phase of the decision frequency of A_k. Phases of frequencies can differ from one agent to the other agent depending on their availability and their time zone. A_k defines decision constraints at each process stage t where $(t - Ph_k)$ value is an integer multiple of $1/F_k$. In order to define a consistent function, we assume that $1/F_k$ is integer.

3.4.2.4 Coefficient of Restriction

Any variable of the design problem can be improved with constraints. This improvement increases the minimum satisfaction and wellbeing of the agent. M_k is the coefficient of restriction for the constraints defined by A_k. This attitude defines the restriction effect of the constraints defined on the solution space. M_k is used as an improvement coefficient for the minimum values of the intervals. The constraint defined by A_k at process stage t is $C_k^t: v_k \geq minv_k^t \times (1 + M_k)$ where v_k can be any variable of the design problem and $minv_k^t$ is its minimum value at process stage *t*. However, $minv_k^t$ value and M_k value should be larger than 0.

If the constraint is consistent for the design model, which means that there is at least one feasible solution after propagating the constraint, then the constraint is accepted. If the constraint is inconsistent, which means that it returns an unfeasible solution space, then it is refused and rejected from the model. The consistency of the constraint depends on the nature of the initial problem and the earlier constraints defined during progress. Thus it depends on the emerging attitudes of the design agents and propagates an uncertainty.

3.4.3 Characterization of Agents

Depending on their attitudes, agents can have different characters. We consider T_k, F_k, M_k attitudes for characterization of A_k. Pr_k is not considered for characterization, because performance values of different agents may not be the same, and they may not have the same unit of measurement. Besides, T_k attitude is defined based on Pr_k, thus it reflects the desires of an agent A_k. Design agents may be more egoistic or more altruistic compared to the others. More egoistic agents try to satisfy their needs at the highest levels without considering other agents. More altruistic agents have an opposite character, taking other agents into consideration. The solution space is shared between design agents, so any restriction performed by an agent on the solution space will decrease the degree of freedom of the other agents and leave less space to them. As Figure 3.6 shows, the reduction of the degree of freedom is on the favorable side of the satisfaction intervals due to the conflicting objectives. Hence, agents with relatively restrictive design attitudes are considered as more egoistic and agents with less restrictive design attitudes are considered as more altruistic.

Figure 3.7 represents egoistic and altruistic characteristics of agents. When two agents are compared, if F_k and M_k attitudes are identical, the agent with the larger T_k is more egoistic than the other, because it will compromise at a higher satisfaction value. Thus, it will restrict the solution space more than the other, until its objective is satisfied. If T_k and M_k are identical, the agent with larger F_k is more egoistic than the other, because when an agent defines decision constraints more frequently, it will restrict the bounded solution space more rapidly during the process. Consequently, it leaves less space to the other agents. If T_k and F_k are identical the agent with larger M_k is more egoistic than the other, because its decision constraints will be more restrictive than the other agent's decision constraints. This will reduce the solution space for the egoistic agent's benefit.

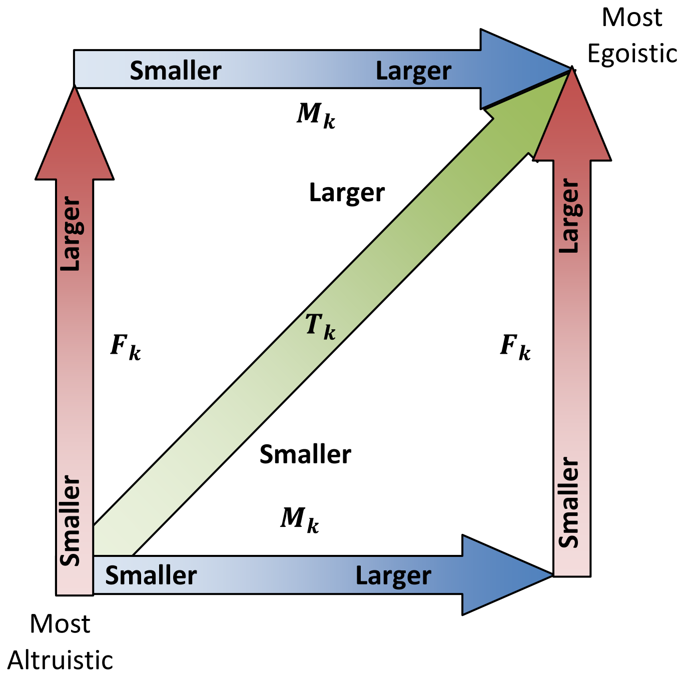


Figure 3.7: Egoism and altruism in design agents

The desires of design agents and their intentions should be rational. If an agent has egoistic desires its intentions are also egoistic. Therefore, more egoistic agents tend to define decision constraints more frequently with a larger coefficient of restriction, and they do not accept to compromise easily. In contrast, more altruistic agents tend to define decision constraints less frequently with a smaller coefficient of restriction, and they can compromise more easily. However the structure of the rationality between the desires and the intentions can be different from agent to agent, since they model human beings. For example, an agent can have a larger M_k value but a smaller F_k value than another agent with the same T_k value. As seen in Figure 3.7, in the extreme case, the most egoistic agent in the design system has the largest T_k, F_k, M_k attitude values while the most altruistic agent has the smallest T_k, F_k, M_k attitude values. The process performance and the design solutions can be influenced by the characters of designers. When the design system consists of heterogeneous agents with different design attitudes, the results may diverge where one agent has a very low satisfaction and another has a very high satisfaction. Process time and the number of conflicts that occur during the design process can also increase due to the non-converging design characters.

3.5 CSP Simulation Process

We present an automatic constraint propagating simulation of our model where the solution space is reduced iteratively considering design agents' BDI mechanism. The objective is to simulate some top-down and bottom-up design processes with different combinations of design agent characters, and compare the results that emerge from these processes. Two practical top-down simulation cases are defined. Case 1 represents the design process where a designer can modify only one design variable after the modification of another designer. Case 2 consists of an all-at-once approach where designers can modify all the design variables after the modification of another designer. Next, two bottom-up simulation cases are defined. In Case 3, designers can modify their design performances. Case 4 is our extended bottom-up design process where designers can modify their wellbeing indicators derived from the performances. In the simulation process we used a split mechanism similar to the round-robin strategy that loops on all the variables at process iteration (Granvilliers, 2012). The objective is to obtain an upper value and a lower value that are as close as possible for each interval. Intervals are reduced until a good degree of precision is obtained. The simulation algorithm is shown in Figure 3.8. We make some assumptions when defining the simulation process:

- If $(t - Ph_k) \times F_k \in \mathrm{Z}$ and $mins_k^t < T_k$ each agent can define (a) decision constraint(s) only once at any iteration and constraints are defined sequentially. If all the agents are processed in iteration then the process passes to the next iteration: t++.

- Decision constraints are defined for improving the worst case scenarios with a coefficient of restriction $M_k > 0$ or $M_{ki} > 0$ or $M_{kj} > 0 \quad \forall k, i, j$. Initial worst cases are larger than 0: $minVn_j^0 > 0\ \forall j$, $minPn_i^0 > 0\ \forall i$, $minwb_k^0 > 0\ \forall k$. If $mins_k^t \geq T_k$ then the compromising agent is extracted from the splitting loop (Cases 1 and 2: $M_{kj} = 0\ \forall j$, Case 3: $M_{ki} = 0\ \forall i$, Case 4: $M_k = 0$).

    - Cases 1 and 2: $Vn_j \geq minVn_j^t \times \left(1 + M_{kj}\right)$ where Vn_j is the normalized design variable *j*, $minVn_j^t$ is its minimum value at iteration *t* and M_{kj} is the coefficient of restriction on the design variable *j* controlled by agent *k*.

    - Case 3: $Pn_i \geq minPn_i^t \times (1 + M_{ki})$ where Pn_i is the normalized performance *i*, $minPn_i^t$ is its minimum value at iteration *t* and M_{ki} is the coefficient of restriction on the performance *i* defined by agent *k*.

    - Case 4: $wb_k \geq minwb_k^t \times (1 + M_k)$.

- If a constraint is rejected, its related coefficient of restriction value is reduced by half. If the coefficient of restriction value of a variable reaches a precision value (*P*), then the splitting is stopped for this variable, because the upper and lower bounds are as close as possible considering the precision value. If all the coefficient of restriction values reach the precision, then the simulation process stops.

- Agents' attitudes are defined at the initial state of the process. M_{ki} and M_{kj} values are equal to M_k at the initial state. Pr_k and T_k attitudes do not change during the simulation process because they represent desires.

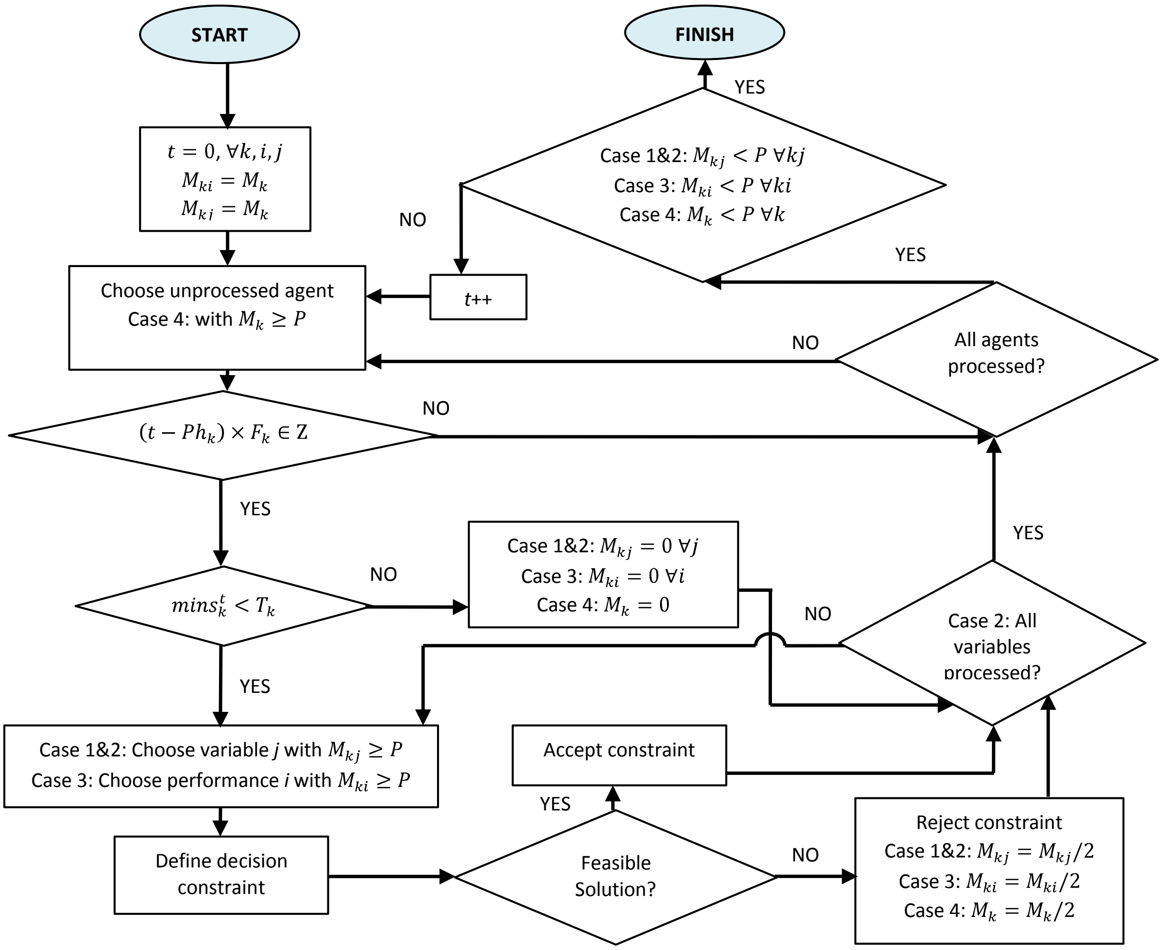

START
FINISH
YES
Case 1&2: $M_{kj} < P\ \forall kj$
Case 3: $M_{ki} < P\ \forall ki$
Case 4: $M_k < P\ \forall k$
NO
$t = 0, \forall k, i, j$
$M_{ki} = M_k$
$M_{kj} = M_k$
YES
Choose unprocessed agent
Case 4: with $M_k \geq P$
t++
NO
All agents processed?
NO
$(t - Ph_k) \times F_k \in \mathrm{Z}$
YES
YES
Case 1&2: $M_{kj} = 0\ \forall j$
Case 3: $M_{ki} = 0\ \forall i$
Case 4: $M_k = 0$
NO
$mins_k^t < T_k$
NO
Case 2: All variables processed?
YES
Case 1&2: Choose variable *j* with $M_{kj} \geq P$
Case 3: Choose performance *i* with $M_{ki} \geq P$
Accept constraint
YES
Define decision constraint
Feasible Solution?
NO
Reject constraint
Case 1&2: $M_{kj} = M_{kj}/2$
Case 3: $M_{ki} = M_{ki}/2$
Case 4: $M_k = M_k/2$

Figure 3.8: CSP simulation algorithm

The simulation process is evaluated by four performance criteria: number of iterations, number of failures, total satisfaction and satisfaction divergence. Four simulation cases are compared regarding these process performances. A smaller number of iterations means a faster convergence of intervals and a rapid design process. This should not however be evaluated alone, because when there are less failures, the coefficient of restriction is split later, which leads to the number of iterations increasing. When a decision constraint is rejected, it is a process failure. Each failure is a potential conflict among designers. Therefore less failures means a design convergence with less conflict. The total number of failures represents the number of potential conflicts occurring in a design process. The process objective is to maximize agents' satisfaction values while minimizing their satisfaction divergence. The satisfaction divergence is defined as the difference between agents' individual satisfactions. In the ideal case, agents should obtain the same satisfaction values, and each value should be equal to 1. Absolute differences of the satisfaction values of either two element combination represent a vector $D(d_1, \ldots, d_n)$. The Euclidian distance of this vector solution to the ideal case solution gives the divergence of the individual satisfaction solutions: $Divergence = \sqrt{(d_1)^2 + \ldots + (d_n)^2}$. More divergent solutions lead to more intense conflicts, because the divergence is caused by agents with a relatively low satisfaction. However, the divergence cannot be evaluated alone. A zero divergence is not desirable if the total satisfaction is zero (all the agents are completely dissatisfied, $s_k = 0\,, \forall k$).

3.6 Monte Carlo Simulation

We ran a Monte Carlo simulation with the design problem of the pressure vessel in (Karandikar and Mistree, 1992; Lewis and Mistree, 1998). We define three agent characters as shown in Table 3.1: Egoistic, Moderate, and Altruistic. We consider that there are no frequency phase differences. All four simulation cases are repeated 1000 times for permutations of these characters generated randomly from their attitude sets. Design agents and their design variables and performances are also chosen randomly in process iterations, so the process sequence is completely independent from agent characters.

Table 3.1: Definitions of random characters

	Egoistic	Moderate	Altruistic
T_k:	(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1)	(0.45, 0.5, 0.55)	(0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4)
F_k:	(0.5, 1)	(1/3, 0.5, 1)	(1/3, 0.5)
M_k:	(5, 6, 7, 8, 9)	(3, 4, 5, 6, 7)	(1, 2, 3, 4, 5)

In the simulation process, the worst cases are improved by increasing the lower bounds of the intervals. Therefore, for minimization objectives the larger bounds are normalized to 0 and the smaller bounds are normalized to 1, and for maximization objectives the smaller bounds are normalized to 0 and the larger bounds are normalized to 1.The precision value is defined as 0.001. This means that if the interval of a variable $X\colon [minX, maxX]$ does not contain $minX \times 1.001$, it is extracted from the loop. Dynamic CSP is defined in C++ computer language and a CP solver library called IBM ILOG CP V1.6 (IBM, 2012) is used to find solutions through its domain reduction and constraint propagation algorithms. The solve function of IBM ILOG CP is used to examine the feasibility of the model.

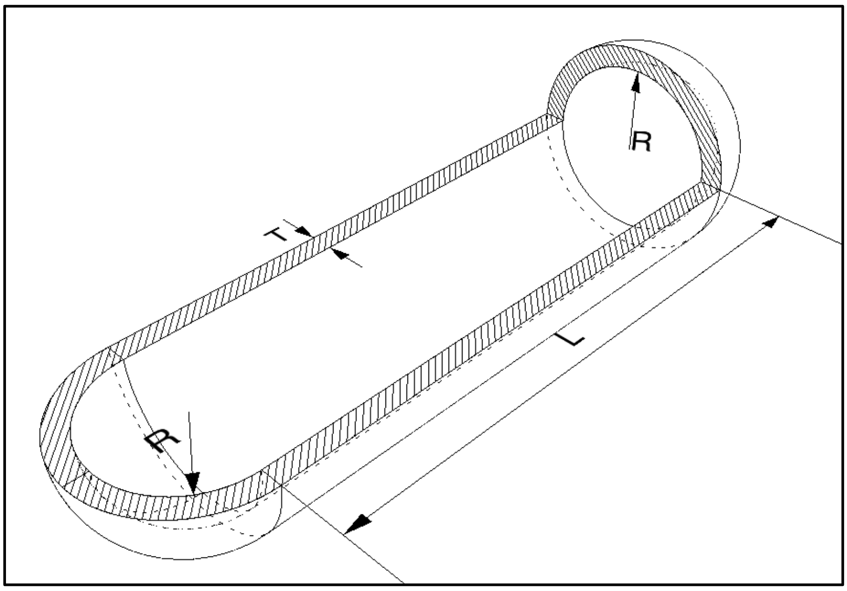


Figure 3.9: Thin-walled pressure vessel

The design problem consists of a cylindrical thin walled pressure vessel with hemispherical ends as shown in Figure 3.9. Problem nomenclature and constant values are given in Table 3.2. There are three design variables (R, T, L) and two design performance variables (W, V). Design performance formulas are given in Table 3.3 and initial constraints are given in Table 3.4. With given constraints, the weight and volume bounds are determined using CP techniques (Table 3.4). The design problem is divided into two sub-problems assigned to two designers (Agent 1 and Agent 2). The objective of Agent 1 is to minimize W by controlling R, T and L while satisfying the related constraints; the objective of Agent 2 is to maximize V by controlling R and L while satisfying the related constraints. Their design activities are coupled because of the shared information in constraints and performance formulas. While Agent 1 minimizes the weight, the volume is minimized; while Agent 2 maximizes the volume, the weight is maximized. Since their objectives are inconsistent, their design activities are conflicting.

Table 3.2: Problem nomenclature and constants

W	Weight of the pressure vessel, lbs.
V	Volume of the pressure vessel, in.^{3}
R	Radius, in.
T	Thickness of the vessel wall, in.
L	Length of the cylinder, in.
P	Pressure inside the cylinder, 3.89 klb.
σ_{UTS}	Ultimate tensile strength of the vessel material, 35 klb.
d	Density of the vessel material, 0.283 lbs./in.^{3}
σ_{circ}	Circumferential stress, lbs./in.^{2}
s_k	Satisfaction of agent *k*
wb_k	Wellbeing of agent *k*
nV	Normalized volume
nW	Normalized weight
nR_k	Radius normalized for agent *k*
nL_k	Length normalized for agent *k*
nT	Normalized thickness
	k=1 for Agent 1 and k=2 for Agent 2

Table 3.3: Design performance formulas

$$W = d\left[\frac{4}{3}\pi(R+T)^3 + \pi(R+T)^2L - \left(\frac{4}{3}\pi R^3 + \pi R^2 L\right)\right]$$

$$V = \frac{4}{3}\pi R^3 + \pi R^2 L$$

Table 3.4: Initial constraints

Stress Constraint:	$\sigma_{circ} = \frac{PR}{T} \le \sigma_{UTS}$
Geometric Constraints:	$5T - R \le 0$
	$R + T - 40 \le 0$
	$L + 2R + 2T - 150 \le 0$
Bounds:	$0.1 \le R \le 36$
	$0.1 \le L \le 140$
	$0.5 \le T \le 6$
	$13.7288 \le W \le 56907.6$
	$67.4138 \le V \le 480404$

Table 3.5: Preferences

If $W \leq 8000\ lbs$ then $s_1 = 1$
If $W \geq 31000\ lbs$ then $s_1 = 0$
If $8000\ lbs < W < 31000\ lbs$ then $1 > s_1 > 0$
If $V \geq 380000\ in.^3$ then $s_2 = 1$
If $V \leq 120000\ in.^3$ then $s_2 = 0$
If $120000\ in.^3 < V < 380000\ in.^3$ then $0 > s_2 > 1$

Table 3.6: Normalizations

$13.7288 \leq W \leq 56907.6$:	$1 \geq nW \geq 0$
$67.4138 \leq V \leq 480404$:	$0 \leq nV \leq 1$
$0.5 \leq T \leq 6$:	$1 \geq nT \geq 0$
$0.1 \leq R \leq 36$:	$1 \geq nR_1 \geq 0$
	$0 \leq nR_2 \leq 1$
$0.1 \leq L \leq 140$:	$1 \geq nL_1 \geq 0$
	$0 \leq nL_2 \leq 1$

Agents define their performance satisfaction functions with piecewise constraints as shown in Table 3.5. All the transitions between the preferences are considered linear as shown in Figure 3.5. Design performances and design variables are normalized using their bound values. Piecewise constraints are defined for normalizations and are shown in Table 3.6. Agent 1 minimizes *W* through minimizing *R, T* and *L* and Agent 2 maximizes *V* through maximizing *R* and *T*. Supplementary initial constraints are defined in order to avoid non-zero worst case scenarios for enabling fruitful constraint propagations ($s_1, s_2, nT, nR_1, nR_2, nL_1, nL_2 \geq 0.01$). These very small constraint values do not affect the performance of the simulation process.

All the permutations of egoistic (E), moderate (M) and altruistic (A) characters of Agent 1 and Agent 2 are simulated for each case 1000 times through a Monte Carlo simulation approach and the average results are shown in Figure 3.10. Case 1, the process which enables modifications on design variables only one agent at a time, requires the longest process time because the number of iterations is the largest for every character combination. One of the bottom-up approaches outperforms the second top-down design approach, Case 2, for every character permutation except EM. When there is at least one altruistic agent, Case 4 outperforms Case 3 except AE.

The process time should be evaluated with the number of failures, because when the number of failures decreases, M_k is split less, and the convergence continues during subsequent process stages. The number of failures can be considered as the number of conflicts. Case 1 and Case 2 result in the highest number of failures. Case 3 and Case 4 generate significantly less conflicts. Case 4 outperforms Case 3 except when one of the agents is moderate and the other is egoistic or both of them are egoistic.

The intensity of the conflicts also needs to be evaluated. A conflict is more reasonable when its intensity is relatively high, because one agent covers more space, resulting in the blocking of the other agent in order to satisfy preferences. When the final wellbeing values are compared (Figure 3.11), it is significant that in Case 1 and Case 2, Agent 1 dominates Agent 2 regardless of their characters. In Case 3, Agent 2 generally dominates Agent 1 except when Agent 1 is more egoistic than Agent 2 (MA and EA). In Case 4, when agents reflect the same characters, no domination occurs, except when one is more egoistic than the other. These findings are reflected in the divergence results. Case 4 generates significantly the least divergence regardless of agent characters. Its conflicts are relatively less intense. Case 4 generates larger total satisfaction than Cases 1 and 2. However compared to Case 3, the reduction of divergence is obtained by slightly compromising some of the total satisfaction value for some character combinations.

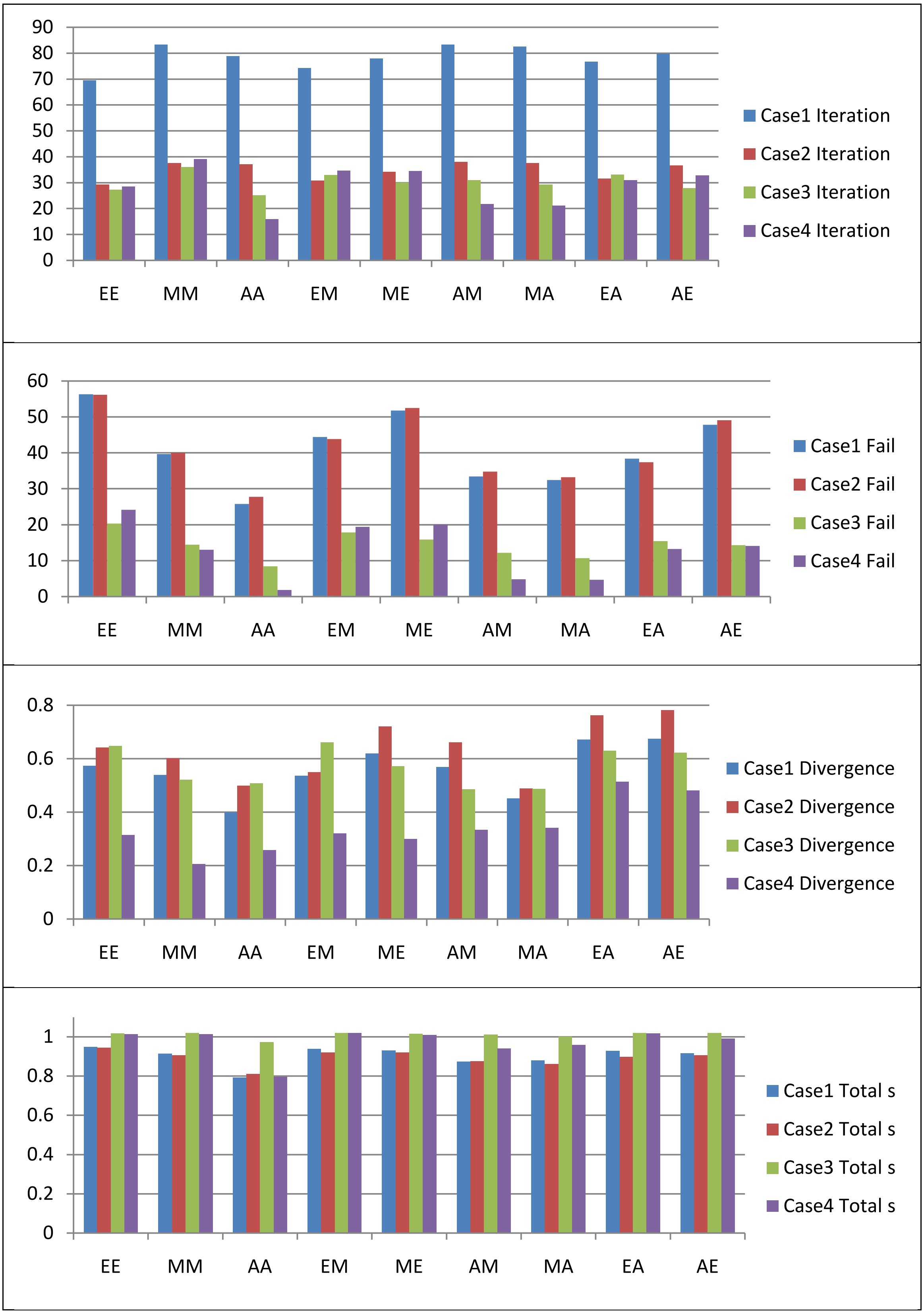


Figure 3.10: Simulation results

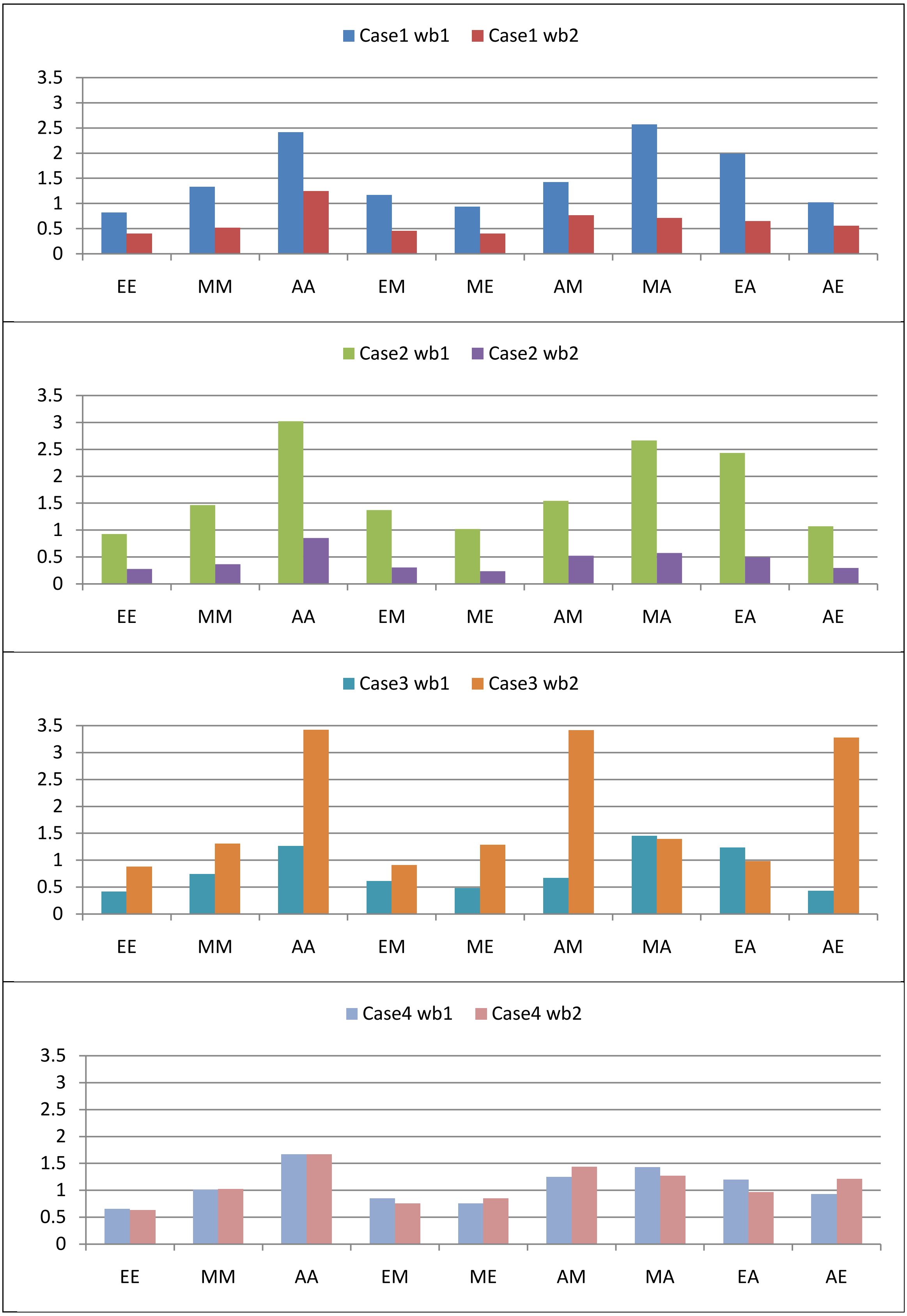


Figure 3.11: Final wellbeing values

Figure 3.12 shows the average total satisfaction values and the average satisfaction divergence values obtained by our approach for Case 4. Optimal results are obtained when both agents are moderate, because the satisfaction divergence is minimal while the total satisfaction value is maximal. Egoistic agents overestimate their desires represented as compromise threshold values when they work with an egoistic agent or a moderate agent, because the total satisfaction values of these situations are not greater than the total satisfaction of the situation where both agents are moderate. Also, the satisfaction divergence values of EE, EM and ME situations are larger than the MM situation. This shows that the individual satisfaction values diverge more because of the egoistic attitudes reflected during the design process. Altruistic agents underestimate their desires when the other agent is altruistic or moderate. Even if altruistic agents can be over-satisfied, the total satisfaction values of AA, AM and MA situations are smaller than the total satisfaction values of the other character situations.

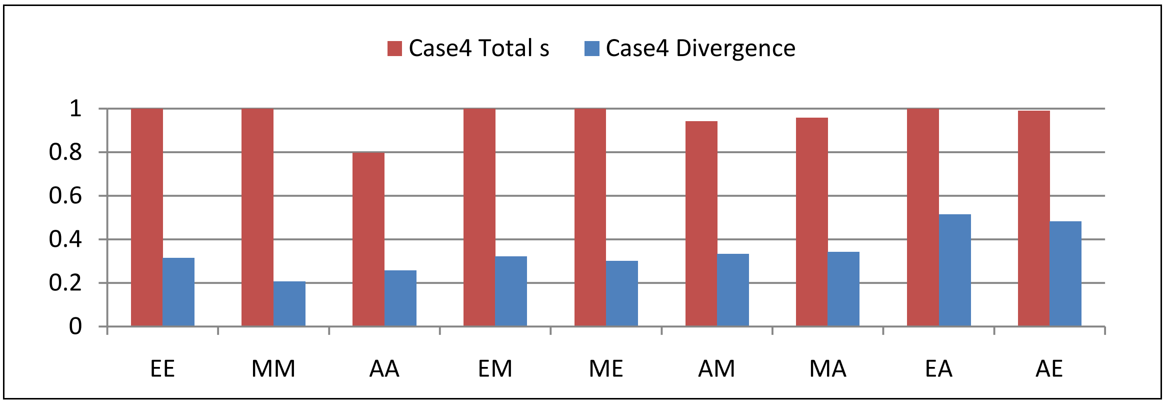


Figure 3.12: Case 4 satisfaction results

3.7 Conclusions

In this paper, we define an extended bottom-up design approach, exploring agent-based attitude modeling techniques within the set-based design concept. The conventional bottom-up design approach is usually defined at problem level; however design attitudes that define beliefs, desires and intentions are overlooked at the initial state of the problem, so trade-offs on design preferences remain abstract. In contrast, our extended bottom-up design approach includes design preferences at an earlier state and explores the solution space with design preferences emerging from the desires of various designers.

We perform a CSP simulation for different designer characters. The simulation results show that when design attitudes of heterogeneous designers in distributed design are not evaluated beforehand, the performance of the design process is significantly lower. Regardless of the designer characters, significant dominations usually occur on the same designer. This means that the results are mostly influenced by the process itself. Consequently, individual solutions do not converge in equilibrium, so conflicts are unavoidable. However, when design attitudes are evaluated beforehand, designers can make trade-off intentions on their wellbeing values derived from their beliefs and desires. With this approach, designer domination is relatively less significant and is coherent with designer characters. This shows that the results are only influenced by the design attitudes. Designers can therefore converge in equilibrium. Consequently, the number of conflicts and the divergence of the solutions that result in the intensity of the conflicts are prevented.

Other conclusions deduced from our approach are about how the satisfaction results emerge from different reciprocal design attitudes. It is shown that reciprocal egoistic attitudes can cause diverging satisfaction results. In contrast, more altruistic reciprocal attitudes can decrease the divergence of individual satisfactions. However, too much altruism can decrease the total satisfaction obtained from the final solution, except when the reciprocating design agent reflects very significant egoistic attitudes.

Our approach is capable of determining and preventing design conflicts, but it does not provide any strategies for resolving existing conflicts. Some cooperative conflict resolution strategies can be defined and integrated into the same platform, but this requires design agents to negotiate, and compromising constraints through relaxing them.

3 References

Antonsson, E.K., Otto, K.N., 1995. Imprecision in engineering design. Journal of Mechanical Design 117, 25–32.

Bazzan, A.L.C., Bordini, R.H., Campbell, J.A., 2002. Evolution of agents with moral sentiments in an iterated prisoner's dilemma exercise, in: Parsons, S., Gmytrasiewicz, P., Wooldridge, M. (Eds.), Game Theory and Decision Theory in Agent-Based Systems, Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer US, pp. 43–64.

Bratman, M.E., Israel, D.J., Pollack, M.E., 1988. Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355.

Caballero, A., Botía, J., Gómez-Skarmeta, A., 2011. Using cognitive agents in social simulations. Engineering Applications of Artificial Intelligence 24, 1098–1109.

Castelfranchi, C., Rosis, F.D., Falcone, R., Pizzutilo, S., 1998. Personality traits and social attitudes in multiagent cooperation. Applied Artificial Intelligence 12, 649–675.

Chanron, V., Lewis, K., 2005. A study of convergence in decentralized design processes. Res Eng Design 16, 133–145.

Cohen, P.R., Levesque, H.J., 1990. Intention is choice with commitment. Artificial Intelligence 42, 213–261.

Devendorf, E., Lewis, K., 2011. The impact of process architecture on equilibrium stability in distributed design. Journal of Mechanical Design 133, 101001.

Fathianathan, M., Panchal, J.H., 2009. Modelling an ongoing design process utilizing top-down and bottom-up design strategies. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 223, 547–560.

Goyal, M., 2005. Attitude based teams in a hostile dynamic world. Knowledge-Based Systems 18, 245–255.

Granvilliers, L., 2012. Adaptive bisection of numerical CSPs, in: Milano, M. (Ed.), Principles and Practice of Constraint Programming, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 290–298.

Huang, H.-Z., Gu, Y.-K., Du, X., 2006. An interactive fuzzy multi-objective optimization method for engineering design. Engineering Applications of Artificial Intelligence 19, 451–460.

IBM, 2012. IBM ILOG CPLEX CP Optimizer for Constraint Programs - Features and benefits [WWW Document]. URL http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/about/

Jennings, N.R., Campos, J.R., 1997. Towards a social level characterisation of socially responsible agents. IEE Proceedings - Software Engineering 144, 11.

Karandikar, H., Mistree, F., 1992. Designing a composite material pressure vessel for manufacture: A case study in concurrent engineering. Engineering Optimization 18, 235–262.

Kim, H.M., Michelena, N.F., Papalambros, P.Y., Jiang, T., 2003. Target Cascading in Optimal System Design. Journal of Mechanical Design 125, 474.

Koulinitch, A.S., Sheremetov, L.B., 1998. Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor. Expert Systems with Applications 15, 295–307.

Kwon, O.B., Lee, K.C., 2002. MACE: multi-agents coordination engine to resolve conflicts among functional units in an enterprise. Expert Systems with Applications 23, 9–21.

Lewis, K., Mistree, F., 1998. Collaborative, sequential, and isolated decisions in design. Journal of Mechanical Design 120, 643.

Lin, R., Kraus, S., Wilkenfeld, J., Barry, J., 2008. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. Artificial Intelligence 172, 823–851.

Malak, R.J., Aughenbaugh, J.M., Paredis, C.J.J., 2009. Multi-attribute utility analysis in set-based conceptual design. Computer-Aided Design 41, 214–227.

Meyer, Y., Yvars, P.-A., 2012. Optimization of a passive structure for active vibration isolation: an interval-computation- and constraint-propagation-based approach. Engineering Optimization 44, 1463–1489.

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

Panchal, J.H., Fernández, M.G., Paredis, C.J.J., Allen, J.K., Mistree, F., 2007. An Interval-based Constraint Satisfaction (IBCS) method for decentralized, collaborative multifunctional design. Concurrent Engineering 15, 309–323.

Papalambros, P.Y., Michelena, N.F., Kikuchi, N., 1997. Distributed cooperative systems design. Proceedings of the 11 th international conference on engineering design 2, 265–270.

Park, H., Michelena, N., Kulkarni, D., Papalambros, P.Y., 2001. Convergence criteria for hierarchical overlapping coordination of linearly constrained convex design problems. Computational Optimization and Applications 18, 273–293.

Parry, G.W., 1996. The characterization of uncertainty in Probabilistic Risk Assessments of complex systems. Reliability Engineering & System Safety 54, 119–126.

Parsons, M.G., Singer, D.J., Sauter, J.A., 1999. A hybrid agent approach for set-based conceptual ship design, in: Proceedings of 10th International Conference on Computer Applications in Shipbuilding. Cambridge, MA.

Pita, M.S., Lima Neto, F.B., 2007. Simulations of egoistic and altruistic behaviors using the vidya multiagent system platform, in: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, GECCO '07. ACM, New York, NY, USA, pp. 2927–2932.

Rodriguez, S., Julián, V., Bajo, J., Carrascosa, C., Botti, V., Corchado, J.M., 2011. Agent-based virtual organization architecture. Engineering Applications of Artificial Intelligence 24, 895–910.

Sobek, D.K., Ward, A.C., Liker, J., 1999. Toyota's principles of set-based concurrent engineering. Sloan Management Review 40, 67–83.

Wang, J., Terpenny, J., 2003. Interactive evolutionary solution synthesis in fuzzy set-based preliminary engineering design. Journal of Intelligent Manufacturing 14, 153–167.

Ward, A.C., Liker, J., Sobek, D.K., Cristiano, J.J., 1994. Set-based concurrent engineering and Toyota, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 79–90.

Wooldridge, M., Jennings, N.R., 1995. Intelligent agents: Theory and practice. Knowledge Engineering Review 10, 115–152.

Xianjia, W., Weibing, L., 2009. Preference and evolution in the iterated prisoner's dilemma. Acta Mathematica Scientia 29, 456–464.

Yannou, B., 2004. Managing uncertainty of product data. An enhancement on constraint programming techniques, in: Tichkiewitch, S., Brissaud, D. (Eds.), Methods and Tools for Co-operative and Integrated Design Methods and Tools for Co-operative and Integrated Design. Kluwer Academic Publishers, Springer, pp. 195–208.

Yannou, B., Harmel, G., 2004. A comparative study of constraint programming techniques over intervals in preliminary design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 189–198.

Yannou, B., Harmel, G., 2006. Use of constraint programming for design, in: ElMaraghy, H.A., ElMaraghy, W.H. (Eds.), Advances in Design, Springer Series in Advanced Manufacturing. Springer London, pp. 145–157.

Yvars, P.-A., 2009. A CSP approach for the network of product lifecycle constraints consistency in a collaborative design context. Engineering Applications of Artificial Intelligence 22, 961–970.

Zhao, L., Jin, Y., 2003. Work structure based collaborative engineering design. ASME Conference Proceedings 865–874.

Zheng, Y., Shen, H., Sun, C., 2011. Collaborative design: Improving efficiency by concurrent execution of Boolean tasks. Expert Systems with Applications 38, 1089–1098.

Chapter 4: Resolving Design Conflicts and Promoting Solidarity in Distributed Design

Baris Canbaz, Bernard Yannou, Pierre-Alain Yvars

This paper has been submitted to *Engineering Applications of Artificial Intelligence.*

Abstract

The resolution of complex design problems requires a distributed design system that considers the involvement of various designers. Inconsistencies of design objectives and working procedures of distributed subsystems can cause design conflicts due to couplings among their sub-problems. Another issue is the management of imprecision in design systems caused by the lack of knowledge about the final decision. In this paper we define a conflict management model using the concept of set-based design (SBD) in order to overcome these issues. We utilize constraint satisfaction problem (CSP) techniques and model agent attitudes to detect and justify design conflicts of heterogeneous design agents. A novel cooperative CSP (CoCSP) is defined for resolving design conflicts through compromising constraint restriction. The conflict resolution system can be adopted for different strategies which take into account the solidarity architecture of design agents. The gains and costs of centralized, decentralized and controlled conflict resolution system strategies are simulated with Monte Carlo simulations where design agent characters and their interactions reflect a stochastic nature.

4.1 Introduction

Engineering design processes of complex products and services require the collaboration of multiple design experts from different disciplines, and these can be located in different places. Concurrent collaborative design activities ensure the feasibility and increase the probability of success of new product development projects by providing necessary expertise and reducing the time to market. Since there are physical divisions between design experts and/or disciplinary boundaries within the multi-disciplinary design problem, a distributed design approach can be adopted

(Sobieszczanski-Sobieski et al., 1984). In distributed design, while the global design problem is decomposed into sub-problems, design responsibility is decentralized and distributed to subsystems composed of one or more design experts (Papalambros et al., 1997). Subsystems have limited control over the design variables because of their limited expertise and responsibility. The ultimate objective of collaborative distributed design is to resolve sub-problems concurrently so that the global multi-objective design problem converges to a global optimum. However, as Lewis and Mistree (1998) point out, in reality it is highly unlikely to obtain true concurrency, because subsystems are not independent, but are related to each other through couplings between their sub-problems. Inconsistencies in the design system can result in design conflicts through couplings. Design conflicts arise during the design process when designers are not able to satisfy their own design objectives. Inconsistencies can be found at both problem level and process level. Problem level inconsistencies consist of non-uniform, in other words conflicting, local design objectives of subsystems. Favoring the design objective of a designer can be detrimental to the design objectives of the other designers. Process level inconsistencies consist of conflicting working procedures of subsystems (Zhao and Jin, 2003). For instance, a designer that influences the design model more frequently and restrictively can block the other designers trying to satisfy their own design objectives. Design conflicts are justified when the satisfaction levels of designers obtained from the global solution diverge, resulting in a situation where a designer is very satisfied and the rest are not satisfied or dissatisfied. This divergence represents the intensity of the design conflicts. Preventing, justifying and resolving design conflicts are indispensable concepts for obtaining globally satisfactory design solutions where satisfaction levels of subsystems are in equilibrium.

Many propositions have been made for design conflict resolution models. The most significant approaches are agent-based models. Klein (1991) proposes a heuristic-based computational model that produces advice for resolving conflicts between design agents by utilizing the knowledge about conflict resolution strategies obtained from empirical design expertise. Wong (1997) proposes a method of cooperative knowledge-based systems that includes a library of multi-agent design conflict resolution strategies that can be combined in an appropriate order for the situation, so that if one strategy fails the system tries the next one. Koulinitch and Sheremetov (1998) define a constraint-based dynamic design system model that includes facilitator agents which send messages to relax some constraints until a consistent solution is obtained. Li et al. (2002) propose an integration-based conflict resolution system that includes a hierarchical constraint network to detect design conflicts. A knowledge-based method, a constraint relaxation method, and a negotiation method are used to resolve various conflicts. Shin et al. (2006) propose a design conflict resolution model that employs agent-based negotiation techniques for facilitating a goal-formation process that generates fuzzy goals and modifies them by coordination between agents. The other significant conflict resolution approaches utilize mathematical optimization techniques and fuzzy logic models. Yin et al. (2008) propose a combinatorial heuristic algorithm for design conflict negotiation which is based on Fuzzy Matter Element Particle Swarm Optimization (FMEPSO). Jin et al. (2009) propose a design conflict resolution algorithm that optimizes the design problem by considering the fuzziness of design variables and the additional cost of conflict resolution. Li et al. (2004) propose a graph model for conflict resolution, not only for design problems, but for all types of conflicting decision making problems of multiple stakeholders. In this approach, the uncertainty of decision maker preferences is modeled, and four types of solution definitions are developed by modeling human behavior under conflict. The graph model is extended with fuzzy preferences by Bashar et al. (2012). Although these various efforts provide improvements to the design system, some important aspects for managing design conflicts are overlooked. Mathematical optimization models overlook the dynamic nature of the design problem that changes with the evolving intentions of designers, reflecting designer reactions to the uncertainty. Agent-based conflict resolution models consider the dynamic interactions of design agents; however they do not consider modeling design attitudes that define reactions and interactions of various design agents. This is an important omission, because modeling design attitudes can help to explore design conflicts. In addition, with the exception of some models that represent design variables and decisions with fuzzy parameters, the imprecision caused by the lack of information about design consequences is largely overlooked. Imprecision is inherent to design problems, as it represents the epistemic uncertainty of what the results from the emerging design interactions might be (Parry, 1996). According to Malak et al. (2009) this issue requires representing the uncertainty with imprecise intervals/sets and delaying uncertain decisions to later process stages where the information about the related decision is available. Besides, the proposed conflict resolution methodologies do not discuss the adoption strategy of the conflict resolution system with regard to the solidarity architecture of the system participants. Centralized conflict resolution system strategies provoke or oblige solidarity between design agents to resolve design conflicts. In contrast, the decentralized conflict resolution system strategy considers an autonomous solidarity where agents are free to decide whether to help to resolve design conflicts. The question of which strategy should be adopted remains unanswered.

In our earlier research (Canbaz et al., 2013), we defined a novel bottom-up design approach that employs the concept of Set-based Design (SBD) for managing imprecision in design and modeling design agent attitudes for exploring design conflicts. It was demonstrated through Monte Carlo simulations that our agent-based SBD approach prevents design conflicts that arise from heterogeneous designer attitudes. In this paper we extend this approach and integrate a conflict management model. In Section 4.2 we discuss the ability of SBD and constraint satisfaction problem (CSP) techniques to manage imprecision in design. Our conflict management model is introduced in Section 4.3 and the CSP simulation process of this model is presented in Section 4.4. Monte Carlo simulations of different conflict resolution systems and the uncooperative design system are performed on a design problem of a multi-disc clutch system that involves variable agent characters. The simulation problem definitions and simulation results that compare different strategies considering their gains and costs are presented in Section 4.5.

4.2 SBD and CSP Techniques

Variables of coupled and conflicting design problems cannot be crisply defined due to the lack of information about the consequences of design decisions (Antonsson and Otto, 1995; Yannou, 2004). The epistemic uncertainty due to this imprecision is very significant especially in preliminary design processes. Deterministic design methods cannot overcome this issue, because they require the restriction of the design problem by attributing crisp values to problem variables so that radical decisions are performed before the information about decisions is certain. Set-based design (SBD) is proposed as an alternative concept which considers the design process as an ongoing evolution of non-crisp concurrent design decisions (Sobek et al., 1999; Ward et al., 1994). Design problem variables are represented as imprecise values in their domains (intervals for real variables), so epistemic uncertainty can be propagated and evaluated. Design decisions related to certain information are performed as constraints on variable domains, so the epistemic uncertainty is reduced. If design decisions are related to uncertain information, they can be delayed to later process stages where more details about the information is gathered due to the reduction of epistemic uncertainty through previous decisions. This design approach provides flexibility of modifications and higher adaptability to changes (McKenney et al., 2011), as well as robustness to design (Parsons et al., 1999). Repetitive design activities and loopbacks are consequently avoided by disclaiming a trial and error approach, so design process time is reduced.

Although SBD originated as a management philosophy for concurrent engineering tasks, recent research has shown that SBD can be adopted at a technical solution level with constraint satisfaction problem (CSP) techniques e.g. (Yannou et al., 2013; Meyer and Yvars, 2012; Panchal et al., 2007; Yannou and Harmel, 2006). A CSP is defined with three groups of sets P = (V,D,C), where V is the set of variables, D is the set of domains that contain the allowable values of variables, and C is the set of constraints that restrict the problem (Montanari, 1974). A multidimensional space is defined by the Cartesian product of variable domains and contains the consistent solutions that respect the problem constraints. A design sub-problem is defined by three main problem elements: design variables that can be controlled by the specific subsystem, design performances that are evaluated by the subsystem, and constraints that must be respected for making design decisions. Some of these constraints form the relation between variables and performances; these, or other, constraints can also form couplings between variables and between performances of other sub-problems. A decomposed design problem can be defined with three spaces through CSP definitions: the design space defined by design variables, the performance space defined by design performance variables, and the solution space that contains both design and performance spaces. Design decisions are represented as constraints restricting the solutions space. When the epistemic uncertainty is reduced through design decisions, the remaining solution space of complex problems can be determined precisely with domain reduction/filtering algorithms of constraint programming (CP) techniques (Yannou and Harmel, 2004). For example, X and Y are integer variables and $Z = X \times Y$. Their domains are $(X) = [20\,,30]$, $D(Y) = [15\,,25]$ so the domain of Z is synthesized $D(Z) = [300\,,750]$. If a constraint is defined $X < Y$ then the inconsistent solutions are filtered, so the domains are reduced. The reduced domains are $D(X) =$

$[20\,,24]$ and $D(Y) = [21\,,25]$. Domain reduction due to a constraint leads to domain reduction of related parameters, so $D(Z) = [420\,,600]$. Domain reduction can function with a bottom-up architecture where constraints can be defined directly on value occurrences. For instance if $Z \leq 480$ the domains of the variables are reduced to $D(X) = [20\,,22]$ and $D(Y) = [21\,,24]$. This is a very effective way of representing preferences in design systems, because it enables decision constraints to be defined directly on design performances and on indicators derived from design performances. Yannou and Harmel (2004) demonstrate that CP techniques can compete with and outperform probabilistic and fuzzy methods on managing imprecision in design.

Some derivatives of CSP are made in order to deal with various artificial intelligence platforms that employ a multi-agent system (MAS). Dynamic CSP (DynCSP) allows constraints to be added to or removed from the problem model; the problem evolves over time with some agent actions which are related to the constraints performed through a process (Dechter and Dechter, 1988). The solution space is restricted with the addition of a constraint, or relaxed by the removal of a constraint. The problem at time stage t is $P^t = \langle P^{t-1}, \Delta \rangle$ where P^{t-1} is the problem defined at the previous stage and $\Delta: P^{t-1} \rightarrow P^t$ is a function that maps the previous problem to the problem at stage t. DynCSP is adequate for MASs that require dynamic negotiation and conflict resolution of interacting agents. Distributed CSP (DisCSP) is proposed to divide the CSP into n sub-CSPs shared to n automated agents A_i: $P = \left(P_{A_i} + \cdots + P_{A_n}\right)$ (Yokoo et al., 1998). Agents resolve their own sub-CSPs concurrently, and then their solutions are unified. In DisCSP, an agent shares information only for loose couplings. This reduces the cost of knowledge transfer and avoids privacy/security problems among agents that may be caused by sharing all the information (Faltings and Yokoo, 2005). This can be an advantage for large but not very densely coupled problems. However, as Salido and Barber (2006) highlight, DisCSP is not suitable when the problem is densely coupled or the number of variables is very high. Dense couplings among design agents can cause conflicts in the problem solving stage which require a large number of message and information transfers. This issue renders DisCSP non-effective for conflicting distributed design problems. Cooperative CSP (CoCSP) is the problem technique defined by Yvars (2010, 2009) for obtaining cooperative solutions in MASs. In CoCSP, if a design agent cannot perform its design activities, the other agents can help this agent by compromising their constraints. The CoCSP definition is suitable for dealing with conflicting distributed design problems, because it enables negotiations and conflict resolutions among design agents dynamically during the design process. CoCSP algorithms surveyed in the literature minimize the number of decision constraints rejected at any stage of the design process. This approach considers only the number of decision constraints, but it does not take into account the amount of the restriction and the satisfaction obtained by constraints. When design conflicts emerge from the interactions of heterogeneous designer characters, this approach is inadequate. When design agents are heterogeneous, they define heterogeneous constraints, so that one constraint of a particular agent can be more restricting than two constraints of another agent. Thus the conflict resolution objective of heterogeneous MAS should not compromise the quantity of constraints; it should instead compromise the restriction of constraints. In order to satisfy these requirements, we define our model as an agent-based SBD model that explores designer attitudes for detecting and justifying design conflicts. A novel CoCSP model is developed for resolving design conflicts through compromising the restriction of constraints.

4.3 Conflict Management Model

In SBD, design variables are represented with imprecise domains/intervals. The analysis of the design space emerging from allowable design variable solutions stimulates design agents to react so as to satisfy their design objectives. Design agents react through defining decision constraints in the design model. The reaction of agents to uncertainties of complex dynamic domains is defined by agent attitudes (Goyal, 2005). The most widely deployed architecture of an agent is the Belief-Desire-Intention (BDI) paradigm developed by Bratman et al. (1988). The character of an agent is the combination of its various autonomous attitudes, and different strategies can be developed through exploring agent characters in order to obtain optimal interactions between heterogeneous agents (Castelfranchi et al., 1998). We use CSP definitions and the BDI paradigm to manage design conflicts of autonomous agents. Figure 4.1 shows our adaptation of the BDI mechanism for a design agent.

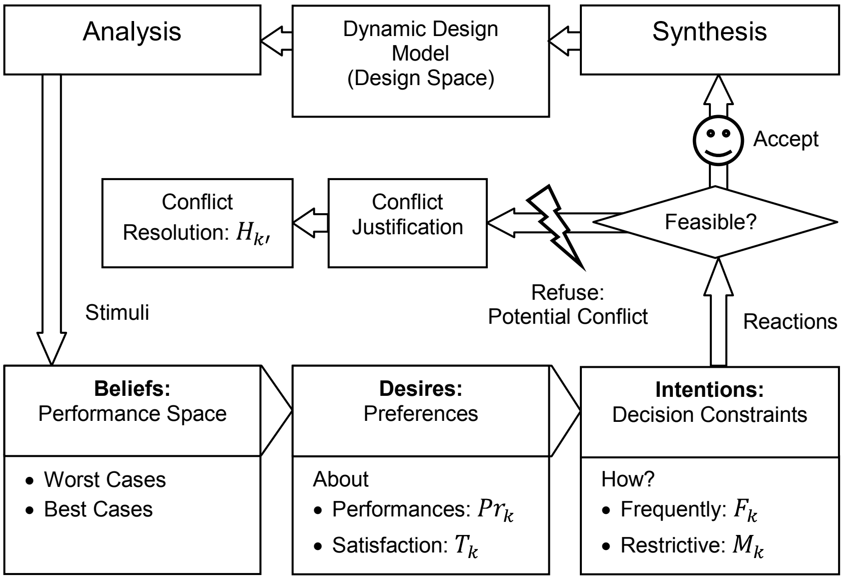


Figure 4.1: BDI Mechanism of Design Agents

We define an agent *k*, A_k, as an entity with five different attitudes: $A_k\langle Pr_k, T_k, F_k, M_k, H_k\rangle$. Analysis of the design space stimulates the design agent, and triggers its BDI mechanism. The design space is transformed through CSP definitions into the performance space where design agents define their design objectives. Upper and lower bounds of the design performance intervals represent the possible worst cases and the possible best cases. These cases reflect the beliefs of the design agent about the convergence of its design performance variables towards its design objectives. The converging intervals of possible worst and possible best cases propagate some uncertainty for the design agent. Reactions of design agents are bound through couplings, because the solution space is shared. If an agent modifies the design model for its own benefits, it can decrease the best case of another agent with a conflicting objective. Thus, the performance intervals of an agent depend on the unpredictable reactions of the other design agents. In order to adopt design performance values, the design agent defines preferences about its design performances and on the emerging satisfaction values from these performances. Preferences reflect the agent's desires towards its uncertain design performances. Pr_k is the set of preferences of the agent on the performance values, and reflects the agent's attitudes for satisfaction obtained from alternative solutions. In a coupled design system, it is highly unlikely to fully satisfy all the design agents. Therefore, design agents are forced to compromise at a certain level on their satisfaction interval. T_k is the compromise threshold value of the agent representing the preference of the agent on the satisfaction values for compromise. Beliefs and desires lead the design agent to state intentions for increasing its satisfaction from the dynamic design model. The design agent reacts by defining decision constraints that restrict the solution space, with the aim of improving its worst cases. Intentions are reflected with how frequently and how restrictively the decision constraints are defined. F_k is the average frequency attitude of the agent for defining constraints in the model. M_k is the coefficient of restriction of constraints defined by the agent, and it reflects the restrictiveness attitude of decision constraints defined by the agent. When a decision constraint is defined, the design model's feasibility is evaluated through testing the consistency of the decision constraint with CP techniques. After the definition of the constraint, the design model is feasible if there is at least one solution remaining in the solution space. Therefore, the decision constraint emerging from the agent's BDI mechanism is accepted and a new design space is synthesized in the dynamic process. The decision constraint is rejected if it yields an empty solution space. When a decision constraint is rejected, it means the design agent could not perform its modifications, so a potential design conflict is detected. The conflict is justified only if the rejection of a decision constraint means an under-satisfied design agent caused potentially by over-satisfaction of another design agent or design agents. This justification process requires CoCSP definitions where all the information about the agents' states and their decision constraints are shared among design agents. If a conflict is justified, the other design agents can help this agent to incorporate its constraint into the design model. H_k is the helping attitude of A_k. If A_k needs help, the other design agents' helping attitudes $H_{k'}$ determine the approval of the conflict resolution process. Conflict justification and resolution models are defined in the following sub-sections.

4.3.1 Conflict Justification

In order to justify conflicts, we evaluate design agents' states during the design process with control indicators called wellbeing indicators. The wellbeing indicators are derived from the desires of the agents reflected on the beliefs of the agents. First we model design performances with satisfaction functions defined by piecewise constraints. Satisfaction functions are scaled between 0 and 1. For example, one objective of agent k is to maximize a performance i; the agent is fully satisfied by a performance value above or equal to P_1 and fully dissatisfied by a performance value below or equal to P_2. It is assumed that there is a linear transition between these two preference values. s_{ki} is the satisfaction value of the agent k obtained by the performance i and v_i is the performance value of the performance i. Then the piecewise constraints are as follows:

If $v_i \geq P_1$, $s_{ki} = 1$ (4.1)

If $v_i \leq P_2$, $s_{ki} = 0$ (4.2)

If $P_1 > v_i > P_2$, $1 > s_{ki} > 0$ (4.3)

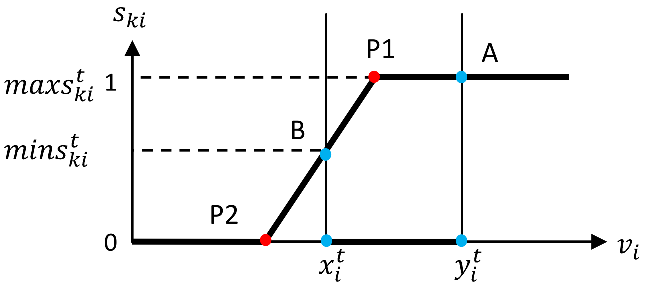


Figure 4.2: Intervals of satisfaction function

SBD process is an ongoing evaluation of intervals, so the design process is divided into stages where design agents make reactions. At process stage *t*, performance *i* is defined with an interval, $[x_i^t , y_i^t]$, where x_i^t is the minimum value and y_i^t is the maximum value. An interval for the satisfaction of agent *k* is obtained from performance *i* at stage *t*: $s_{ki} = [mins_{ki}^t , maxs_{ki}^t]$ where $mins_{ki}^t$ is the minimum satisfaction and $maxs_{ki}^t$ is the maximum satisfaction obtained within the interval $[x_i^t , y_i^t]$. Figure 4.2 demonstrates an example with the piecewise constraints given above. Minimum satisfaction is obtained at point B and maximum satisfaction is obtained between *P1* and point A. If design agents evaluate several design performances, they can assign weights to their satisfaction values according to the importance of the design performances for their job. Therefore individual performance satisfaction values are aggregated, so general satisfaction states of agents can be observed. w_{ki} is the weight assigned to the performance *i* by the design actor *k*. *I* is the total number of the performances considered by design actor k. General satisfaction of an agent k is an interval $s_k = [mins_k^t\,, maxs_k^t]$. Its bounds are calculated with the following equations:

$$mins_k^t = \sum_{i=1}^{I} w_{ki} \times mins_{ki}^t \tag{4.4}$$

$$maxs_k^t = \sum_{i=1}^{I} w_{ki} \times maxs_{ki}^t \tag{4.5}$$

$$\sum_{i=1}^{I} w_{ki} = 1 \;\; \forall k \tag{4.6}$$

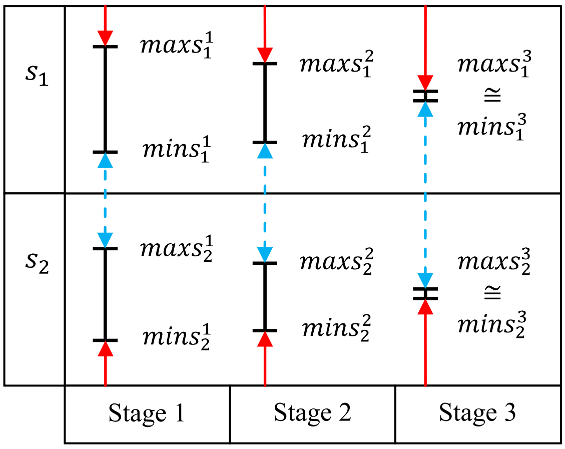


Figure 4.3: Bilateral convergence

Design agents define decision constraints in order to improve their minimum satisfaction values during process stages. However the convergence of their satisfaction intervals is bilateral because of conflicting couplings. Figure 4.3 shows an example where the constraints of Agent 1 are represented with dashed arrows and the constraints of Agent 2 are represented with solid arrows. When a constraint is defined by an agent, it increases the minimum satisfaction value of this agent, but it decreases the maximum satisfaction of the other agent. Therefore at the end of the design process, satisfaction intervals converge to a compromised point solution where minimum and maximum satisfaction values are approximately equal. This solution is uncertain during the design process until it is obtained. Design agents can reflect an attitude of desiring a satisfaction value in which they may compromise. The preference on the satisfaction value is called compromise threshold T_k, which defines the compromise attitude of an agent. The objective of a design agent is to guarantee that its satisfaction interval will converge to a value at least as good as its compromise threshold value. If the minimum satisfaction of an agent reaches its T_k value or passes beyond, then the agent passes to the compromise state. In the compromise state, the agent stops adding decision constraints to the model, so this leaves space to the other agents in the solution space. Satisfaction values are normalized by dividing them by the compromise threshold value and this provides wellbeing states Eq. (4.7, 4.8). Wellbeing is defined with an interval $wb_k = [minwb_k^t \, , maxwb_k^t]$ where minimum value is the minimum wellbeing indicator and maximum value is the maximum wellbeing indicator. Global states of design agents are observed with their wellbeing indicators. If an agent suffers because of design conflicts, it will be detected immediately. If $minwb_k^t$ value is larger than or equal to 1, then agent k is in a perfect wellbeing state. However if agent k is not in a perfect wellbeing state, it defines a decision constraint in order to improve its wellbeing state. If the constraint is rejected because it does not provide any consistent solution, and if there is at least one agent k' in a better wellbeing state with a higher $minwb_{k'}^t$, then the design conflict is justified. This is because it is considered that the shared solution space is not restricted in equilibrium. At least one agent restricted the solution space for its benefits more than the suffering agent that cannot get its decision constraint accepted. The suffering agent can therefore ask help to the other agents in better wellbeing states in order to resolve the justified design conflict. If the conflict is not justified, then the agent does not deserve the conflict resolution. This agent must reduce its M_k value in order to define a less restrictive constraint at the following design process stage.

$$minwb_k^t = \frac{mins_k^t}{T_k} \qquad (4.7)$$

$$maxwb_k^t = \frac{maxs_k^t}{T_k} \qquad (4.8)$$

4.3.2 Conflict Resolution

Design agents intend to improve their minimum wellbeing states. Intentions are represented with how frequently and how restrictively their constraints are defined. F_k is the average frequency of agent k to define its decision constraints in the model. Ph_k is the phase of the decision frequency of A_k. In internationally distributed design systems, agents are available in different time zones, so phases of frequencies can be different from one agent to another. Agent k defines decision constraints at each process stage *t* where $(t - Ph_k)$ value is an integer multiple of $1/F_k$. We consider that $1/F_k$ is an integer number, because process stages are represented with integer values. M_k is the coefficient of restriction for the constraints defined by agent k in order to improve its wellbeing state.

SBD is an on-going restriction of the solution space, so we assume that agents restrict their wellbeing intervals by defining increasingly restrictive constraints. The constraint defined by agent *k* at process stage *t* is $C_k^t\colon wb_k \geq minwb_k^t \times (1 + M_k)$ where $minwb_k^t$ value and M_k value are larger than 0. We assume that F_k is a fixed value during the process, because it represents the average. However M_k value can change during the process depending on how restrictively the agent intends to define its constraint at the process stage.

Both F_k and M_k define the working procedures of agent *k.* Since design agents are autonomous, their design attitudes reflected during the design process can be heterogeneous. Inconsistencies can arise among heterogeneous working procedures through design couplings. We explore design agents' working procedures to resolve design conflicts. When a decision constraint of an agent k is accepted, it is put on a list of accepted constraints L_k. When a constraint defined by an agent is not consistent, then the constraint is refused because it causes an unfeasible solution space. Other agents can help to enable this constraint by removing some of their constraints from their list of accepted constraints. In our CoCSP, we assume that only one agent can offer to cooperate at any one time: multiple agents do not cooperate.

Our conflict resolution model can detect which agent can help and how it can help optimally. The model is composed of three phases. The first phase is the negotiation phase where we detect all the help possibilities. The second phase is the testing phase where the feasibilities of different help possibilities are tested with CP techniques and the optimal help among the feasible help solutions is detected. The third phase is the approval phase where we detect if the help is approved or not by the helping agent.

4.3.2.1 Negotiation Phase

When agents are asked to help, they negotiate the results of the help through comparing their states in order to decide whether they are able to help or not. We assume that agents that are asked to help would want to keep their wellbeing states at least as good as the other agent that needs help after the help is performed. If the wellbeing state of the agent which is asked to help were to go below the wellbeing state of the agent that needs help, then the help is refused. This refusal is reasonable because otherwise the wellbeing state of the helping agent would become inferior after the help, thus generating another conflict.

Figure 4.4 shows an example representing this phenomenon. Here, agents' constraints defined during process stages 0 to 2 and their wellbeing states emerging from these constraints are shown. While all the constraints defined by Agent 2, Agent 3, Agent 4 and Agent 5 are accepted, the constraint C_1^2 of Agent 1 defined at process stage 2 is inconsistent (it yields an unfeasible solution space), so it is refused. Agent 1 needs help in order to enable its constraint. The dashed line compares agents' states. Agent 5 is not able to help, because even without removing any constraint, its wellbeing state would be inferior to the wellbeing state of Agent 1. Agent 2 is not able to help, because in the case of help, it would remove C_2^2 and its wellbeing state would go below the wellbeing state of Agent 1. Agent 3 is able to help through removing C_3^2 or both C_3^1 and C_3^2 without making its wellbeing state inferior to the wellbeing state of Agent 1. Agent 4 is able to help through removing only C_4^2. All the help possibilities are detected through the negotiation phase following this procedure.

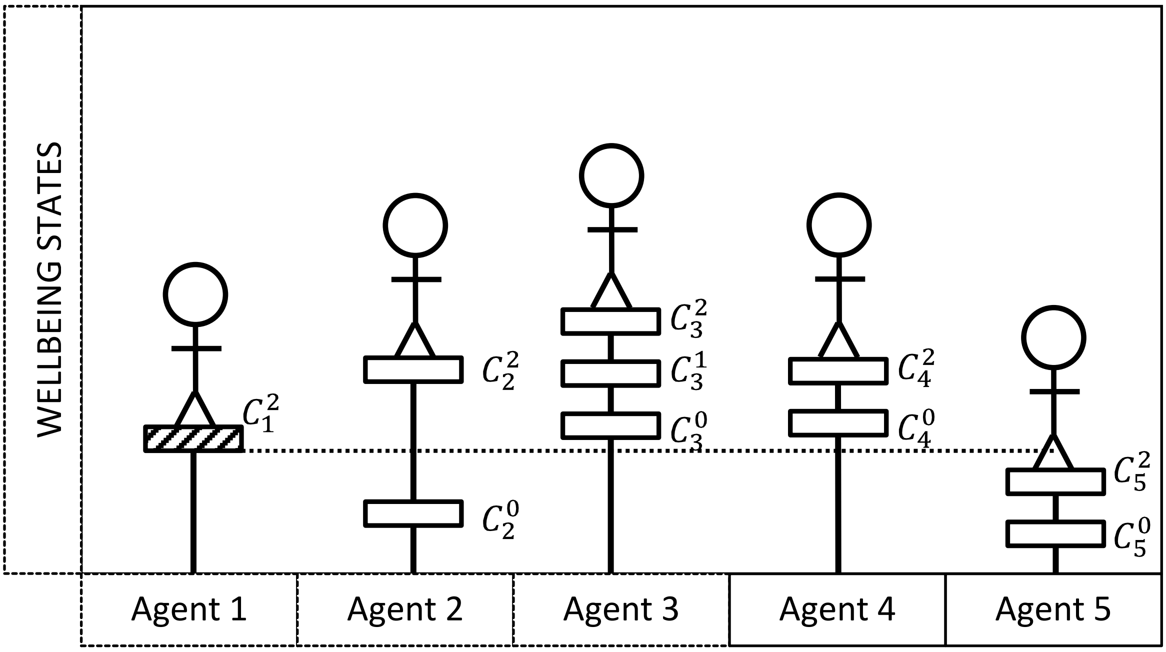


Figure 4.4: Negotiation phase of design agents

4.3.2.2 Testing Phase

If help is possible, its feasibility is tested. Help is feasible if it enables the constraint of the agent that asks for help. This test is performed with CP techniques; the help is feasible if the solution space contains at least one consistent solution after the definition of the conflicting constraint and the removal of the constraint or constraints of the agent that is able to help. Figure 4.5 shows a numerical example where there are three design agents with heterogeneous design attitudes: Agent 1: $\langle T_1 = 0.9\,, F_1 = 0.5\,, M_1 = 0.5\,, H_1 = 0{,}75\rangle$, Agent 2: $\langle T_2 = 0{,}9\,, F_2 = 1\,, M_2 = 0.2\,, H_2 = 0.7\rangle$ and Agent 3: $\langle T_3 = 0.8\,,\ F_3 = 0.5\,,\ M_3 = 0.4\,,\ H_3 = 0.8\rangle$. Agent 2 and Agent 3 start at stage 0 while $Ph_1 = 1$. The constraints defined by agents at process stages, as well as agents' minimum wellbeing values emerging from these constraints are shown. At stage 5, minimum wellbeing values of Agent 1 and Agent 2 are both equal to 1. These agents compromise, so they will not define a constraint during subsequent process stages. All the constraints defined by agents till stage 6 are accepted, so $L_1\colon (C_1^1, C_1^3, C_1^5)$, $L_2\colon (C_2^0, C_2^1, C_2^2, C_2^3, C_2^4, C_2^5)$ and $L_3\colon (C_3^0, C_3^2, C_3^4)$. However, the constraint defined by Agent 3 at stage 6 is refused because it is inconsistent, so it returns an unfeasible solution space. The design conflict is justified, because at stage 5, the wellbeing states of Agent 1 and Agent 2 are better than the wellbeing state of Agent 3. In order to resolve this justified conflict, we detect all the help possibilities that can be provided from Agent 1 and Agent 2. Agent 1 can remove only C_1^5 and Agent 2 can remove only C_2^5 or both C_2^4 and C_2^5. However, Agent 1 refuses to remove constraint combinations that include the (C_1^3, C_1^5) set and Agent 2 refuses to remove constraint combinations that include the (C_2^3, C_2^4, C_2^5) set, because the emerging wellbeing states would fall below the wellbeing state of Agent 3. Next, feasibilities of all the possible helps are tested. If the removal of a constraint combination enables the acceptance of C_3^6, then its help is feasible. Conflict resolution process is unfruitful if there is no feasible help solution. Then Agent 3 reduces M_3 in order to define a less restrictive constraint at the following process stage. If there is more than one feasible help, then we detect the optimal help. We choose the feasible help that gives the maximal $\sum_{k=0}^{K} minwb_k^t$ value after the removal of the constraint or constraints.

	Agent 1	Agent 2	Agent 3
Initial	$minwb_1 = 0.31$	$minwb_2 = 0.34$	$minwb_3 = 0.17$
t=0		$C_2^0: wb_2 \geq 0.41$	$C_3^0: wb_3 \geq 0.24$
t=1	$C_1^1: wb_1 \geq 0.47$	$C_2^1: wb_2 \geq 0.49$	
t=2		$C_2^2: wb_2 \geq 0.59$	$C_3^2: wb_3 \geq 0.33$
t=3	$C_1^3: wb_1 \geq 0.7$	$C_2^3: wb_2 \geq 0.71$	
t=4		$C_2^4: wb_2 \geq 0.85$	$C_3^4: wb_3 \geq 0.47$
t=5	$C_1^5: wb_1 \geq 1$	$C_2^5: wb_2 \geq 1$	
t=6			$C_3^6: wb_3 \geq 0.65$

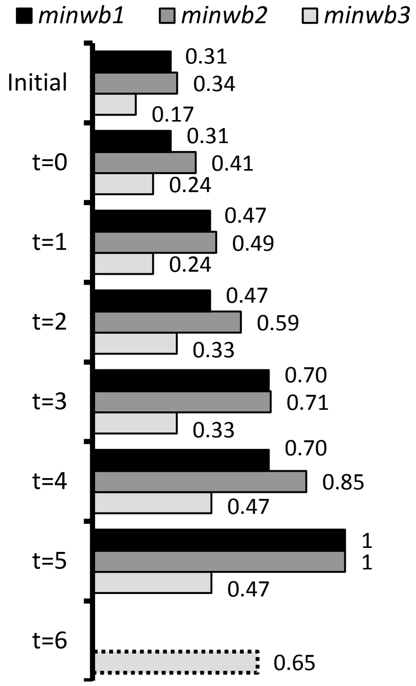


Figure 4.5: Numerical example of conflict management

4.3.2.3 Approval Phase

When the optimal help is detected, our model sends a message to the agent that needs help about who can help, and another message to the agent that can help about exactly how it can help. If the conflict resolution system is decentralized, then helping is under the responsibility of agents. Design agents are autonomous when deciding whether to cooperate by approving the help, or not cooperate by rejecting the help. This is defined by agents' helping attitude H_k. H_k is the probability of agent *k* to approve help. For the example defined in Figure 4.5, the probability of Agent 1 to approve help is 0.75 and the probability of Agent 2 to approve help is 0.7. If the help is approved and performed, then the compromised constraints of the helping agent are removed from the design model and from its list of accepted constraints L_k. The helping agent returns to its wellbeing value and M_k value of the process stage where the remaining most restrictive constraint is defined after the constraint removal. If help is approved by an agent that is in the compromise state, then the agent leaves the compromise state after the help is performed, because its minimum wellbeing value goes below 1. If the help is not approved, the conflict resolution process is unfruitful. Then the agent *k* that asked for help reduces its M_k value in order to define a less restrictive constraint at the following process stage.

When the conflict resolution system is decentralized, agents can seek revenge. For instance, suppose agent *k* needs help and agent *k'* does not approve the help. If at a following process stage agent *k'* needs help and agent *k* can help, then agent *k* seeks revenge by not approving the help to agent *k'* regardless of its H_k. Different control strategies can be defined to encourage design agents to approve the help, or penalize design agents that do not approve the help. Alternatively, a completely centralized conflict resolution system can be adopted where design agents are obliged to approve the help regardless of their helping attitudes.

4.4 CSP Simulation Process

In this section, we present an automatic constraint propagating simulation where the solution space is reduced iteratively considering design agents' BDI mechanism. The objective of our simulation is to evaluate gains and costs of our cooperative conflict resolution model whether the conflict resolution system is centralized or decentralized.

Four different system strategies are defined considering the extent of promotion of solidarity.

- Strategy 1: Uncooperative design system. Agents do not share information about their wellbeing states and constraints, so the design system does not include the conflict management system. If a design conflict arises, it remains unresolved.

- Strategy 2: Decentralized conflict resolution system. Agents share all the information. If a design conflict arises, agents are free to decide whether to cooperate by approving the help, or not cooperate by rejecting the help. Therefore, agents can seek revenge if the help is not approved.

- Strategy 3: Controlled conflict resolution system. Agents share all the information. If a design conflict arises, agents are free to decide whether to cooperate by approving the help, or not cooperate by rejecting the help. However, if an agent does not approve the help, it is penalized by a control agent. A penalized agent cannot define a decision constraint at the next process stage where it is available to define a constraint. After the penalized process stage, it can continue to define decision constraints. Agents do not intend to seek revenge, because uncooperative agents have already been penalized.

- Strategy 4: Centralized conflict resolution. If a design conflict arises, agents are obliged to cooperate by approving the help.

For the simulation we consider process stages as iterations. In the CSP simulation process, we use a split mechanism similar to the round-robin strategy that loops on all the variables at process iteration (Granvilliers, 2012). The objective of this mechanism is to obtain upper and lower values which are as close as possible for each interval. Wellbeing intervals are restricted until a good degree of precision is obtained. The simulation algorithm is shown in Figure 4.6.

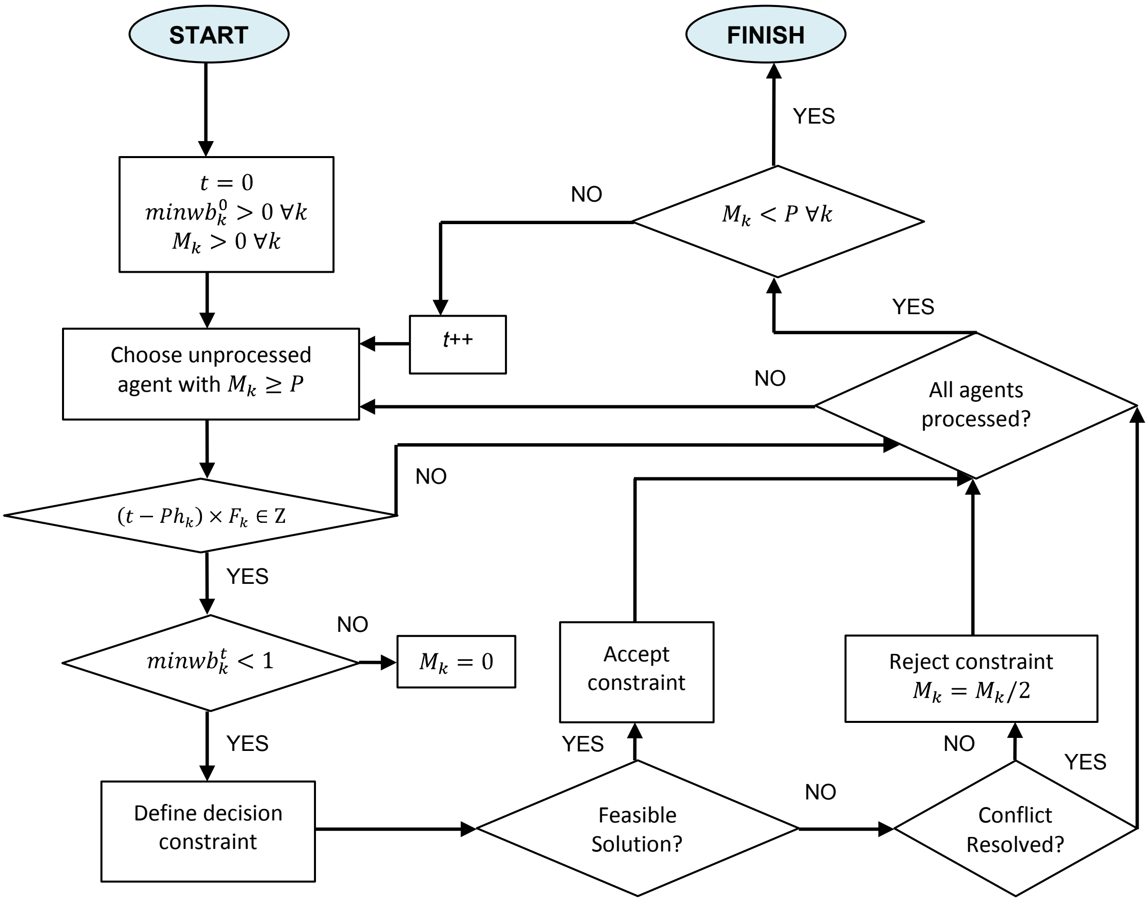


Figure 4.6: Simulation algorithm

We make the following assumptions when defining the simulation process:

- The attitudes of agents are defined at the initial state of the process. Pr_k, T_k and F_k attitudes do not change during the simulation process because Pr_k and T_k represent fixed desires and F_k represents an average value. However, M_k attitude value can change during the process depending on the restrictiveness intentions.

- If $(t - Ph_k) \times F_k \in \mathrm{Z}$ and $minwb_k^t < 1$, each agent k can define a decision constraint only once at any iteration, and constraints are defined sequentially. If $minwb_k^t \geq 1$, then the compromising agent is extracted from the splitting loop with $M_k = 0$. If all the agents are processed in iteration, then the process passes to the next iteration: t++.

- Initial worst cases are larger than 0: $minwb_k^0 > 0\ \forall k$. Decision constraints are defined for improving the worst case scenarios with a coefficient of restriction $M_k > 0\ \forall k$. The constraint defined at process stage *t* by agent k is $C_k^t{:}\, wb_k \geq minwb_k^t \times (1 + M_k)$. We assume that design agents do not restrict their wellbeing intervals, so that minimum wellbeing value surpasses 1. If $minwb_k^t \times (1 + M_k) > 1$, then $1 + M_k = 1/minwb_k^t$.

- If a constraint is rejected and the conflict is not resolved, its related coefficient of restriction value is reduced by half: $M_k = M_k \times 0.5$. Thus, a less restrictive constraint can be defined at the next iteration. If the coefficient of restriction value of an agent reaches a precision value (*P*), then the splitting is stopped for this agent, because the upper and lower bounds of its wellbeing variable are as close as possible considering the precision value. If all the coefficient of restriction values reach P, then the simulation process stops.

- If help is approved, the helping agent returns to the M_k value of the process iteration where the most restrictive constraint is defined after the constraint removal.

The simulation process evaluates the strategies by three process performances: number of iterations, total satisfaction and satisfaction divergence. A smaller number of iterations means a faster convergence of intervals and a rapid design process. In addition, the global objective of the design system is to maximize the satisfaction values of agents while minimizing their divergence. This divergence is defined as the difference between agents' individual satisfaction levels. It represents the degree of intensity of the unresolved design conflicts. In the ideal case, agents should obtain the same satisfaction values and each one should be equal to 1. Absolute differences of the satisfaction values of each two element combination represent a vector $D(d_1, \ldots, d_n)$. The Euclidian distance of this vector solution to the ideal case solution gives the divergence of the individual solutions: $Divergence = \sqrt{(d_1)^2 + \ldots + (d_n)^2}$. More divergent solutions lead to more intense conflicts. However, the divergence cannot be evaluated alone. It should be evaluated with the total satisfaction value, because a zero divergence is not desirable if total satisfaction is zero.

4.5 Monte Carlo Simulation

Design agents can reflect different characters emerging from their BDI mechanism, and this defines an uncertain design system composed of either heterogeneous or homogeneous design agents. Agents' characters can be classed in two extreme groups, namely egoistic characters and altruistic characters (Pita and Lima Neto, 2007). Egoistic agents are motivated by self-interested gains, while altruistic agents are motivated by the benefit of the group that it belongs to.

According to our BDI definition, a design agent $A_k\langle Pr_k, T_k, F_k, M_k, H_k\rangle$ is relatively egoistic if its T_k, F_k, and M_k attitude values are relatively larger, because it desires to compromise at a higher wellbeing state and it intends to define more restrictive constraints more frequently. Its H_k attitude value is also smaller, because it does not intend to help easily. In contrast, an altruistic agent has relatively smaller T_k, F_k, M_k attitude values and a larger H_k value.

The stochastic nature of the system is considered with a Monte Carlo simulation approach. Three different agent characters are defined as shown in Table 4.1: Egoistic, Moderate, and Altruistic. We consider that there are no phase differences of frequencies. Each of the four system strategies is repeated 1000 times with randomly generated agent characters. The same series of random seed numbers is utilized for each strategy, so the simulation results of four strategies are comparable. Also design agents are randomly processed in iterations, so the process sequence is completely independent from agent characters.

The precision value is defined as 0.01. This means that if the interval of wb_k does not contain $minwb_k^t \times 1.01$, it is extracted from the loop at iteration *t*. CSP is defined in C++ computer language and a CP solver library called IBM ILOG CP V1.6 (IBM, 2012) is used to detect consistent solutions precisely through its domain reduction and constraint propagation algorithms. The solve function of IBM ILOG CP is performed to examine the feasibility of the model.

Table 4.1: Definitions of random characters

	Egoistic	Moderate	Altruistic
T_k:	(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1)	(0.45, 0.5, 0.55)	(0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4)
F_k:	(0.5, 1)	(1/3, 0.5, 1)	(1/3, 0.5)
M_k:	(5, 6, 7, 8, 9)	(3, 4, 5, 6, 7)	(1, 2, 3, 4, 5)
H_k	0.2	0.5	0.8

The simulation problem is derived from the example studied in work (Yannou et al., 2010). It is a design problem of a multi-disc clutch system that connects a weight lifter with an engine, followed by a gearbox. This is a complex and a realistic design problem which contains 81 variables and 64 initial constraints. The problem is distributed to four design agents. The design objectives are shown in Table 4.2. This problem is presented in Section 2.5.1. More details of the problem can be seen in Appendix A.

Table 4.2: Problem objectives

	Agent 1	Agent 2	Agent 3	Agent 4
Objectives	Max (m_{load})	Min (m_s)	Min (T)	Max (S_1, S_2, S_3, S_p)

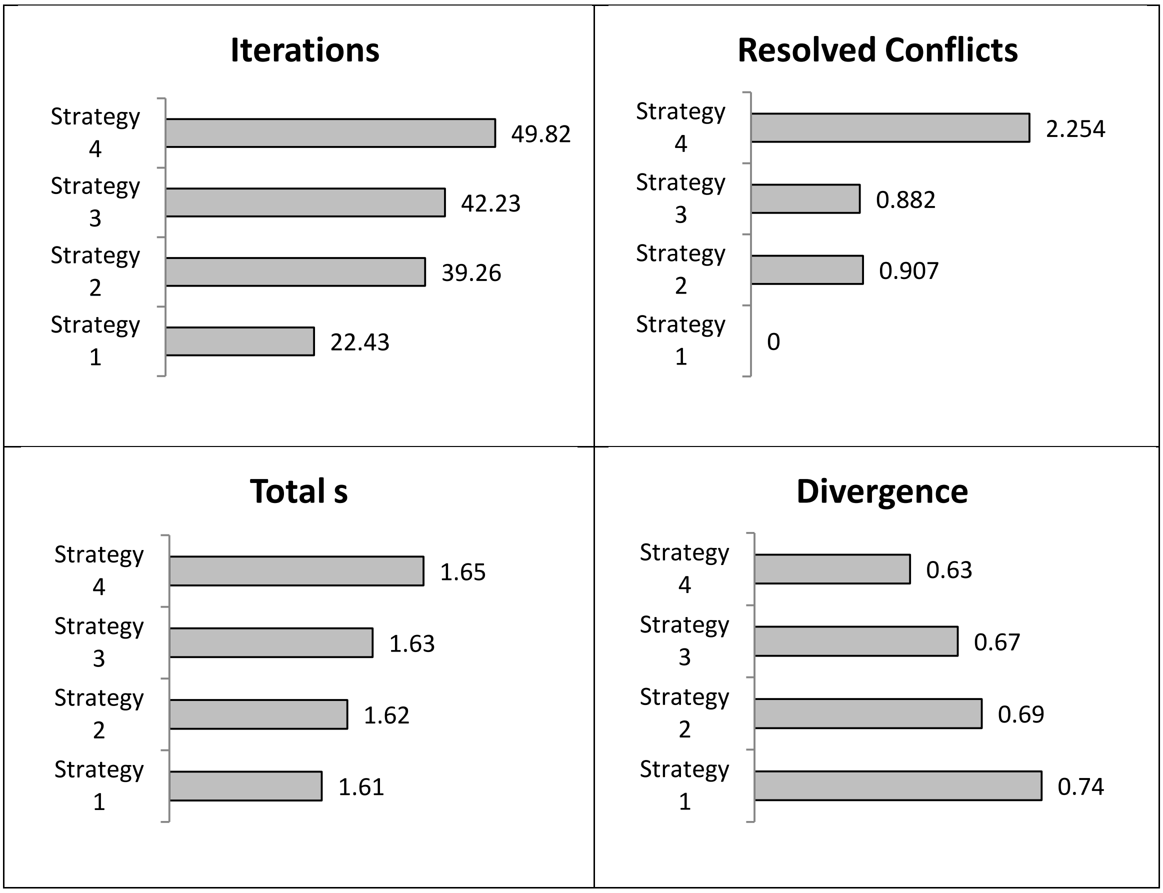


Figure 4.7: Simulation Results

The average results of the Monte Carlo simulation are shown in Figure 4.7. Conflict resolution systems, regardless of their adoption strategy, result in a larger total satisfaction value and a smaller divergence than the uncooperative design system. This shows that conflict resolution systems approach the ideal objective more closely than the uncooperative design system. However, this is obtained at a cost of longer process time. The most rapid system strategy is the uncooperative design system strategy, because there is no conflict resolution that can cause loopbacks to the helping agents. In contrast, the centralized conflict resolution system resolves more design conflicts than the other strategies. This causes more loopbacks, which explains why this strategy generates the longest process time.

The results prove that the divergence is reduced and the total satisfaction is increased with the number of design conflicts resolved or prevented. The least divergence and the greatest total satisfaction values are obtained with the centralized conflict resolution system. These results are the closest to the ideal solution ($s_k = 1, \forall k$; zero divergence, greatest total satisfaction). The controlled conflict resolution system produces better total satisfaction and divergence results than the decentralized conflict resolution system. Even if the number of resolved conflicts of the former is slightly less, the controlled conflict resolution system performance is better, because it prevents some design conflicts by stopping the egoistic actions of agents.

4.6 Conclusions

In this paper, we explored design conflicts with a BDI model and CSP definitions, so that design conflicts can be justified. We defined a CoCSP which is able to manage conflicts by allowing design agents to help others through compromising the restrictiveness of their decision constraints. Different strategies can be adopted for the conflict resolution system. We defined three different conflict resolution system strategies and compared them with each other and with the uncooperative design system that does not include any conflict resolution.

Monte Carlo simulation results show that, regardless of the conflict resolution system strategy adopted, our conflict management model represents a significant improvement over the uncooperative design system. The divergence of individual satisfaction solutions is lowered and the total satisfaction is increased. Thus, through our conflict management model, the design solution tends towards the ideal solution where design agents are completely and equally satisfied in equilibrium. However, this gain is obtained at the cost of the increase of design process time, because conflict resolution causes loopbacks.

Other conclusions are deduced by comparing different adoption strategies of the conflict resolution system. A centralized conflict resolution system strategy can be adopted if the process time is not an important issue and if the main objective is the highest possible degree of conflict resolution. With the lowest divergence and the highest total satisfaction, this system converges to the ideal solution closer than any other system. However, if process time is an important issue, a controlled system where uncooperative agents are penalized can be preferred. If the conflict resolution system is completely decentralized without any control, the solutions are less close to the ideal solution than the centralized and controlled systems.

We conclude that the centralization of the conflict resolution system, either with a control mechanism (penalty) or a complete centralized mechanism (obligation), is more fruitful and should be preferred to complete decentralization. Consequently, informing design agents of their respective situation in terms of their wellbeing – the information transparency value – and encouraging or forcing them to help each other – the solidarity value – are two values we believe efficient for the quality of the resulting design, albeit sometimes to the detriment of design process time.

4 References

Antonsson, E.K., Otto, K.N., 1995. Imprecision in Engineering Design. Journal of Mechanical Design 117, 25–32.

Bashar, M.A., Kilgour, D.M., Hipel, K.W., 2012. Fuzzy Preferences in the Graph Model for Conflict Resolution. IEEE Transactions on Fuzzy Systems 20, 760–770.

Bratman, M.E., Israel, D.J., Pollack, M.E., 1988. Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355.

Canbaz, B., Yannou, B., Yvars, P.-A., 2013. Expanding the bottom-up design approach through integrating design attitudes into set-based design, in: Proceedings of ASME Design Engineering Technical Conferences.

Castelfranchi, C., Rosis, F.D., Falcone, R., Pizzutilo, S., 1998. Personality Traits and Social Attitudes in Multiagent Cooperation. Applied Artificial Intelligence 12, 649–675.

Dechter, R., Dechter, A., 1988. Belief Maintenance in Dynamic Constraint Networks, in: American Association for Artificial Intelligence. Presented at the 6th National Conference on Artificial Intelligence (AAAI-88), pp. 37–42.

Faltings, B., Yokoo, M., 2005. Introduction: Special Issue on Distributed Constraint Satisfaction. Artificial Intelligence 161, 1–5.

Goyal, M., 2005. Attitude based teams in a hostile dynamic world. Knowledge-Based Systems 18, 245–255.

Granvilliers, L., 2012. Adaptive Bisection of Numerical CSPs, in: Milano, M. (Ed.), Principles and Practice of Constraint Programming, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 290–298.

IBM, 2012. IBM ILOG CPLEX CP Optimizer for Constraint Programs - Features and benefits [WWW Document]. URL http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/about/

Jin, G., Ying, B., Rong, Z., 2009. A conflict resolution algorithm considering design objective optimization in collaborative design, in: 16th International Conference on Industrial Engineering and Engineering Management, 2009. IE EM '09. Presented at the 16th International Conference on Industrial Engineering and Engineering Management, 2009. IE EM '09, pp. 1669–1674.

Klein, M., 1991. Supporting conflict resolution in cooperative design systems. IEEE Transactions on Systems, Man and Cybernetics 21, 1379–1390.

Koulinitch, A.S., Sheremetov, L.B., 1998. Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor. Expert Systems with Applications 15, 295–307.

Lewis, K., Mistree, F., 1998. Collaborative, Sequential, and Isolated Decisions in Design. Journal of Mechanical Design 120, 643.

Li, K.W., Hipel, K.W., Kilgour, D.M., Fang, L., 2004. Preference uncertainty in the graph model for conflict resolution. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 34, 507–520.

Li, X., Zhou, X., Ruan, X., 2002. Conflict management in closely coupled collaborative design system. International Journal of Computer Integrated Manufacturing 15, 345–352.

Malak, R.J., Aughenbaugh, J.M., Paredis, C.J.J., 2009. Multi-attribute utility analysis in set-based conceptual design. Computer-Aided Design 41, 214–227.

McKenney, T.A., Kemink, L.F., Singer, D.J., 2011. Adapting to Changes in Design Requirements Using Set-Based Design. Naval Engineers Journal 123, 66–77.

Meyer, Y., Yvars, P.-A., 2012. Optimization of a passive structure for active vibration isolation: an interval-computation- and constraint-propagation-based approach. Engineering Optimization 44, 1463–1489.

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

Panchal, J.H., Fernández, M.G., Paredis, C.J.J., Allen, J.K., Mistree, F., 2007. An Interval-based Constraint Satisfaction (IBCS) Method for Decentralized, Collaborative Multifunctional Design. Concurrent Engineering 15, 309–323.

Papalambros, P.Y., Michelena, N.F., Kikuchi, N., 1997. Distributed Cooperative Systems Design, in: Proceedings of the 11th International Conference on Engineering Design. Tampere, Finland, pp. 265–270.

Parry, G.W., 1996. The characterization of uncertainty in Probabilistic Risk Assessments of complex systems. Reliability Engineering & System Safety 54, 119–126.

Parsons, M.G., Singer, D.J., Sauter, J.A., 1999. A hybrid agent approach for set-based conceptual ship design, in: Proceedings of 10th International Conference on Computer Applications in Shipbuilding. Cambridge, MA.

Pita, M.S., Lima Neto, F.B., 2007. Simulations of egoistic and altruistic behaviors using the vidya multiagent system platform, in: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, GECCO ’07. ACM, New York, NY, USA, pp. 2927–2932.

Salido, M.A., Barber, F., 2006. Distributed CSPs by graph partitioning. Applied Mathematics and Computation 183, 491–498.

Shin, M., Cha, Y., Ryu, K., Jung, M., 2006. Conflict detection and resolution for goal formation in the fractal manufacturing system. International Journal of Production Research 44, 447–465.

Sobek, D.K., Ward, A.C., Liker, J., 1999. Toyota’s Principles of Set-Based Concurrent Engineering. Sloan Management Review 40, 67–83.

Sobieszczanski-Sobieski, J., Barthelemy, J.F.M., Giles, G.L., 1984. Aerospace engineering design by systematic decomposition and multilevel optimization, in: 14th Congr. of the International Council of the Aeronautical Sciences (ICAS). Toulouse, France.

Ward, A.C., Liker, J., Sobek, D.K., Cristiano, J.J., 1994. Set-based concurrent engineering and Toyota, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 79–90.

Wong, S.T.C., 1997. Coping with conflict in cooperative knowledge-based systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 27, 57–72.

Yannou, B., 2004. Managing uncertainty of product data. An enhancement on constraint programming techniques, in: Tichkiewitch, S., Brissaud, D. (Eds.), Methods and Tools for Co-operative and Integrated Design Methods and Tools for Co-operative and Integrated Design. Kluwer Academic Publishers, Springer, pp. 195–208.

Yannou, B., Harmel, G., 2004. A Comparative Study of Constraint Programming Techniques Over Intervals in Preliminary Design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 189–198.

Yannou, B., Harmel, G., 2006. Use of Constraint Programming for Design, in: ElMaraghy, H.A., ElMaraghy, W.H. (Eds.), Advances in Design, Springer Series in Advanced Manufacturing. Springer London, pp. 145–157.

Yannou, B., Mazur, C., Yvars, P.-A., 2010. Parameterization and dimensioning of the multi-disc clutch in the CO4 environment, Technical report (Cahier d'Etudes et de Recherches), Ecole Centrale Paris, September 7th 2010, available at “http://www.lgi.ecp.fr/uploads/Recherche/CO4Multi_DiscClutch_Technical_Document” on July 9th 2013.

Yannou, B., Yvars, P.-A., Hoyle, C., Chen, W., 2013. Set-based design by simulation of usage scenario coverage. Journal of Engineering Design. DOI:10.1080/09544828.2013.780201.

Yin, Y., Sun, L., Guo, C., 2008. A policy of conflict negotiation based on fuzzy matter element particle swarm optimization in distributed collaborative creative design. Computer-Aided Design 40, 1009–1014.

Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K., 1998. The distributed constraint satisfaction problem: formalization and algorithms. IEEE Transactions on Knowledge and Data Engineering 10, 673–685.

Yvars, P.-A., 2009. A CSP approach for the network of product lifecycle constraints consistency in a collaborative design context. Engineering Applications of Artificial Intelligence 22, 961–970.

Yvars, P.-A., 2010. A Constraint Based Decision Support System for Deadlocks Resolution in Collaborative New Product Design. Journal of Decision Systems 19, 57–74.

Zhao, L., Jin, Y., 2003. Work Structure Based Collaborative Engineering Design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 865–874.

Chapter 5: Experimenting the Conflict Management Model with Human Agents

In this chapter, the conflict management model presented earlier in Chapter 4 is tested and the experimentation results are exposed. In this experimentation, computer agents simulating human behavior are replaced by real human agents. For this purpose, a Serious Game transformation is performed through and adapted user interface.

5.1 Introduction

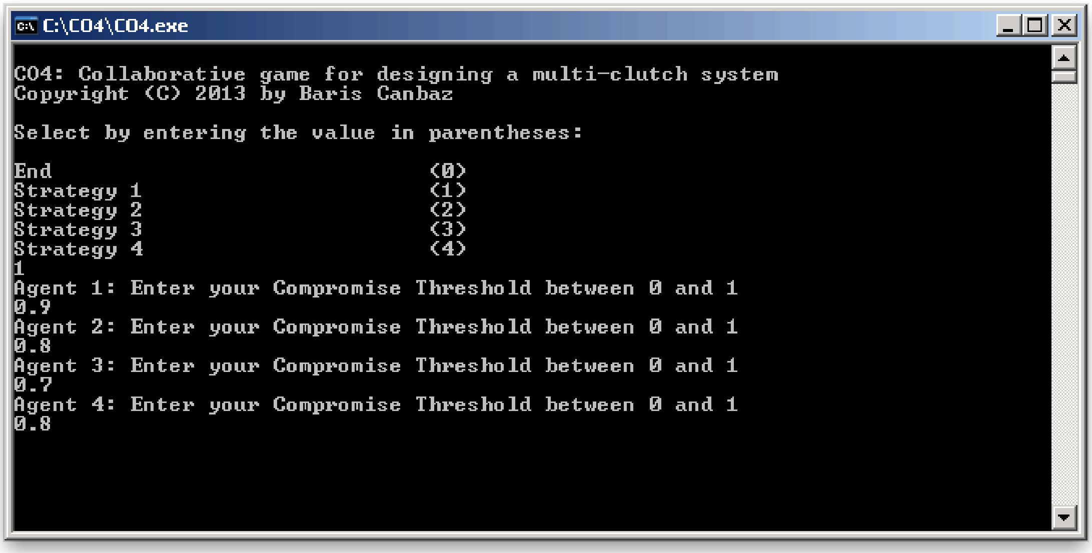
The experimentation objective is to test the conflict management model through human interactions in order to demonstrate its final application and validate the results of Chapter 4. Four adoption strategies defined in Section 4.4 are experimented. The experimentation problem is the design problem of the multi-disc clutch system. It is derived from the example studied in work (Yannou et al., 2010). This problem is already presented in Section 2.5.1. More details of the problem are presented in Appendix A. It is a complex problem containing 81 variables and 64 initial constraints, and it requires deep knowledge about mechanics. In order to avoid training human agents in mechanics about the meaning of variables, constraints and objectives, an analogy of the design problem is built and the design actors are represented by roles in a Serious Game approach.

Serious Games are developed for simulating complex problems with players that do not have proper expertise about the problem (Djaouti et al., 2011). It requires the transformation of the complex problem into a more entertaining one while maintaining its primary purpose (Marfisi-Schottman et al., 2010). The clutch problem is transformed into a problem where the limited resources of a university are shared between four students for scholarships. While players negotiate for increasing their scholarships for an internship in a foreign country, they indeed design the multi-disc clutch system. The Serious Game transformation is presented in Appendix B. In the designing problem of multi-disc clutch system, four designers are required. In its serious game, each sub-problem is a role associated with a human agent. In order to obtain comparable results from experiments of different adoption strategies, we make some considerations:

- Each adoption strategy is experimented with the same group of human agents.
- Human agents maintain their roles when the strategy is changed.
- We make a hypothesis: a human agent is an entity that reflects always the same character (e.g. egoistic, altruistic). By accepting this hypothesis, we consider that characters of human agents are not biased by the adoption strategy and by the order of the adoption strategy.

5.2 Text-based User Interface

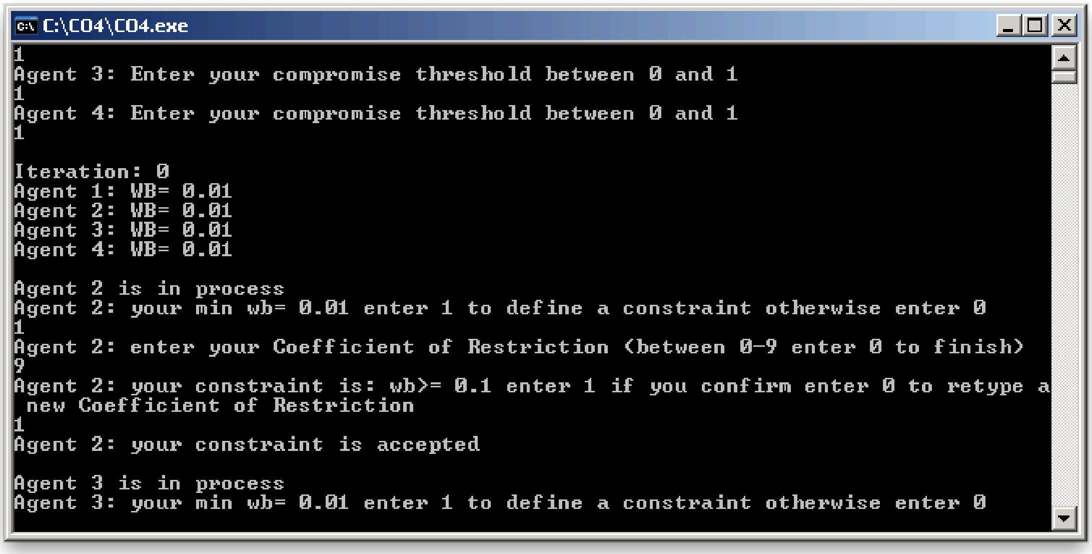
A Text-based User Interface (TUI) is developed for the Serious Game. Each agent has a dedicated screen where he/she can read the instructions, messages and wellbeing information and react by typing numbers. Different stages of the TUI are shown in Figures 5.1-5.10. The initial screen of the TUI is shown in Figure 5.1. Firstly, human agents chose together the adoption strategy among four strategies. Next each agent defines his/her compromise threshold value between 0 and 1. In Strategy 1, agents do not share their screens, while in Strategies 2, 3, and 4 they share their screens. Thus, in Strategies 2, 3, and 4 agents are informed about the other agents' wellbeing values, actions (whether they define a constraint or not), constraint definitions and constraint inconsistencies (whether the constraint is accepted or rejected).



```
C04: Collaborative game for designing a multi-clutch system
Copyright (C) 2013 by Baris Canbaz

Select by entering the value in parentheses:

End                              (0)
Strategy 1                       (1)
Strategy 2                       (2)
Strategy 3                       (3)
Strategy 4                       (4)
1
Agent 1: Enter your Compromise Threshold between 0 and 1
0.9
Agent 2: Enter your Compromise Threshold between 0 and 1
0.8
Agent 3: Enter your Compromise Threshold between 0 and 1
0.7
Agent 4: Enter your Compromise Threshold between 0 and 1
0.8
```

Figure 5.1: Initial screen

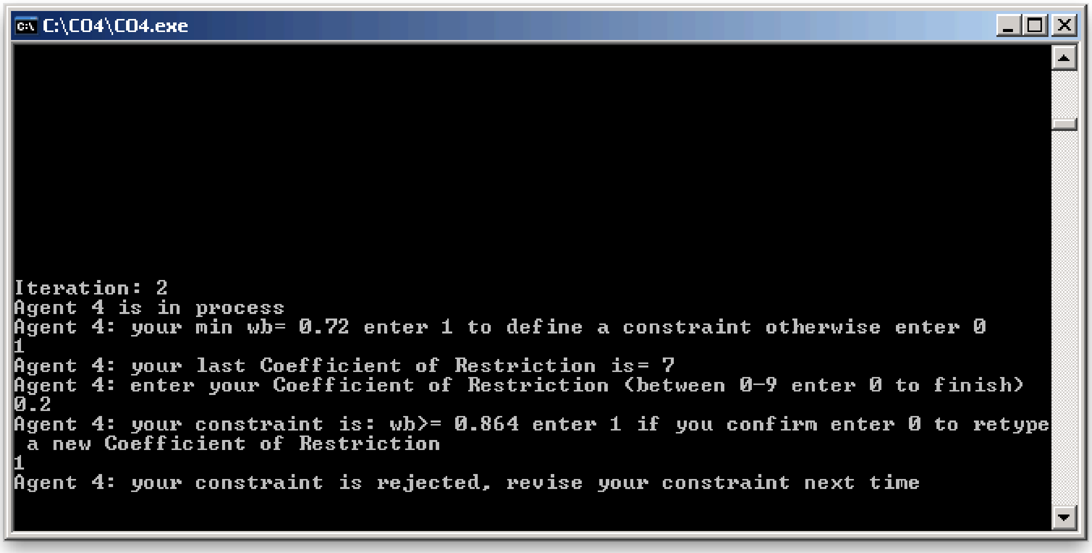
The game consists of iterations where agents react sequentially. Agents are processed randomly at process iteration. If all the four agents are processed, then the process passes to the next iteration. Agents' wellbeing variables are represented as intervals. As shown in Figure 5.2, the agent in process (agent k) is firstly informed about his/her minimum wellbeing at process stage t ($minwb_k^t$), and asked whether to define a constraint or not. If the agent chooses to not to define a constraint, then the next agent is processed. If the agent k chooses to define a constraint in order to improve his/her wellbeing (wb_k), then a coefficient of restriction value (M_k) is requested. The coefficient of restriction can be defined between 0 and 9. The constraint is defined by $wb_k \geq minwb_k^t \times (1 + M_k)$. In this game, we consider that $minwb_k^t$ does not surpass 1. If $minwb_k^t \times (1 + M_k) > 1$, then $1 + M_k = 1/minwb_k^t$. The agent can confirm this constraint or retype another coefficient of restriction if he/she wants to modify the constraint.



```
1
Agent 3: Enter your compromise threshold between 0 and 1
1
Agent 4: Enter your compromise threshold between 0 and 1
1

Iteration: 0
Agent 1: WB= 0.01
Agent 2: WB= 0.01
Agent 3: WB= 0.01
Agent 4: WB= 0.01

Agent 2 is in process
Agent 2: your min wb= 0.01 enter 1 to define a constraint otherwise enter 0
1
Agent 2: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
9
Agent 2: your constraint is: wb>= 0.1 enter 1 if you confirm enter 0 to retype a
 new Coefficient of Restriction
1
Agent 2: your constraint is accepted

Agent 3 is in process
Agent 3: your min wb= 0.01 enter 1 to define a constraint otherwise enter 0
```

Figure 5.2: Constraint definition and acceptance

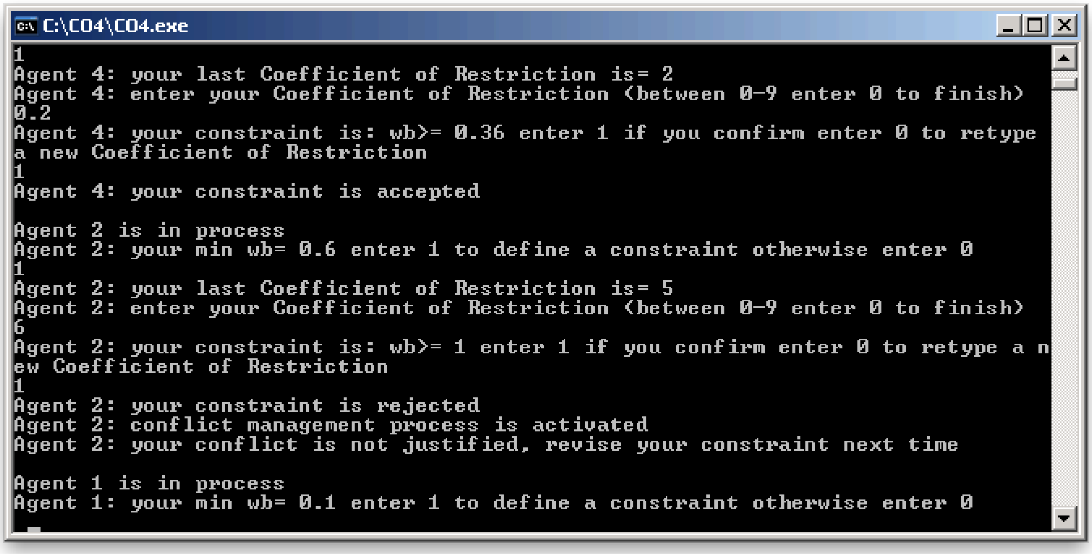
The consistency of the constraint is tested by IBM ILOG CP (2012), and the agent is reported about whether his/her constraint is accepted (shown in Figure 5.2), or rejected (shown in Figures 5.3-5.8). If the constraint is accepted, the process passes to the next agent. If the constraint is rejected, the constraint management process is activated in Strategies 2, 3, and 4. In Strategy 1, there is no conflict management. As shown in Figure 5.3, if the constraint is rejected in Strategy 1, then the agent should revise the constraint at next iteration in order to make it less restrictive.



```
Iteration: 2
Agent 4 is in process
Agent 4: your min wb= 0.72 enter 1 to define a constraint otherwise enter 0
1
Agent 4: your last Coefficient of Restriction is= 7
Agent 4: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
0.2
Agent 4: your constraint is: wb>= 0.864 enter 1 if you confirm enter 0 to retype
 a new Coefficient of Restriction
1
Agent 4: your constraint is rejected, revise your constraint next time
```

Figure 5.3: Constraint definition and rejection in Strategy 1

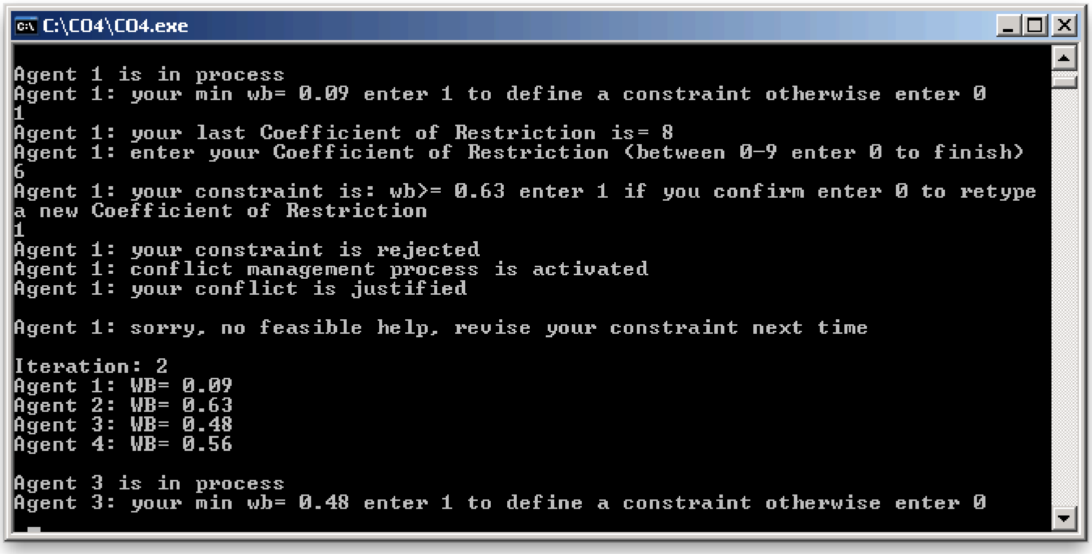
In further iterations, agents are also informed about their last coefficient of restriction defined. Thus, they can track their previous reactions. This can be seen in Figure 5.3. In Strategies 2, 3 and 4, agents are informed about whether their conflict is justified or not. If the conflict is not justified as shown in Figures 5.4, then the agent should revise the constraint at next iteration in order to make it less restrictive.



```
1
Agent 4: your last Coefficient of Restriction is= 2
Agent 4: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
0.2
Agent 4: your constraint is: wb>= 0.36 enter 1 if you confirm enter 0 to retype
a new Coefficient of Restriction
1
Agent 4: your constraint is accepted

Agent 2 is in process
Agent 2: your min wb= 0.6 enter 1 to define a constraint otherwise enter 0
1
Agent 2: your last Coefficient of Restriction is= 5
Agent 2: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
6
Agent 2: your constraint is: wb>= 1 enter 1 if you confirm enter 0 to retype a n
ew Coefficient of Restriction
1
Agent 2: your constraint is rejected
Agent 2: conflict management process is activated
Agent 2: your conflict is not justified, revise your constraint next time

Agent 1 is in process
Agent 1: your min wb= 0.1 enter 1 to define a constraint otherwise enter 0
```

Figure 5.4: Report of unjustified conflict

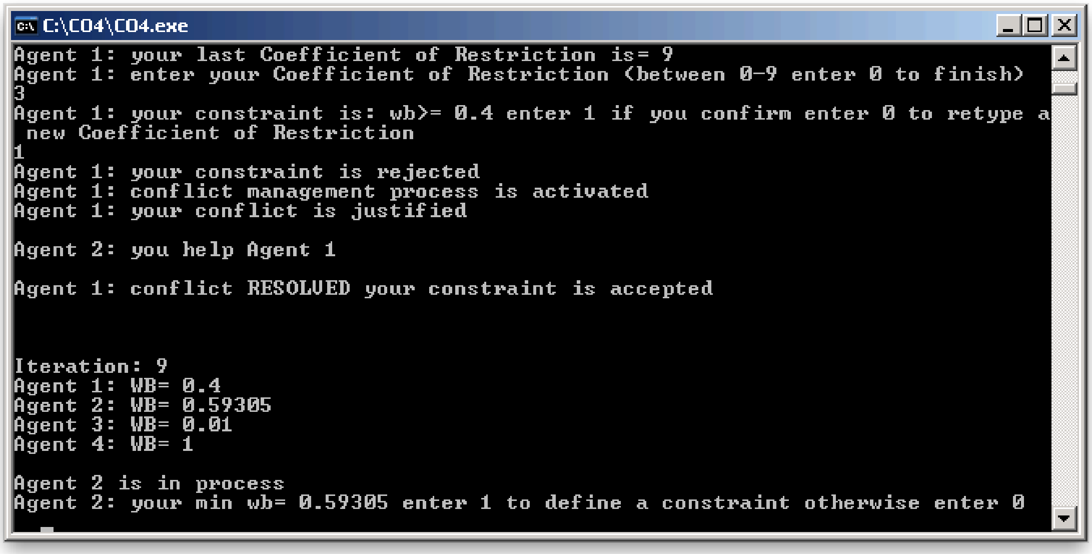
If the conflict is justified, the agent is reported about whether there is a feasible help or not. If there is no feasible help as shown in Figure 5.5, the agent should revise the constraint at next iteration.



```
Agent 1 is in process
Agent 1: your min wb= 0.09 enter 1 to define a constraint otherwise enter 0
1
Agent 1: your last Coefficient of Restriction is= 8
Agent 1: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
6
Agent 1: your constraint is: wb>= 0.63 enter 1 if you confirm enter 0 to retype
a new Coefficient of Restriction
1
Agent 1: your constraint is rejected
Agent 1: conflict management process is activated
Agent 1: your conflict is justified

Agent 1: sorry, no feasible help, revise your constraint next time

Iteration: 2
Agent 1: WB= 0.09
Agent 2: WB= 0.63
Agent 3: WB= 0.48
Agent 4: WB= 0.56

Agent 3 is in process
Agent 3: your min wb= 0.48 enter 1 to define a constraint otherwise enter 0
```

Figure 5.5: Report of no feasible help

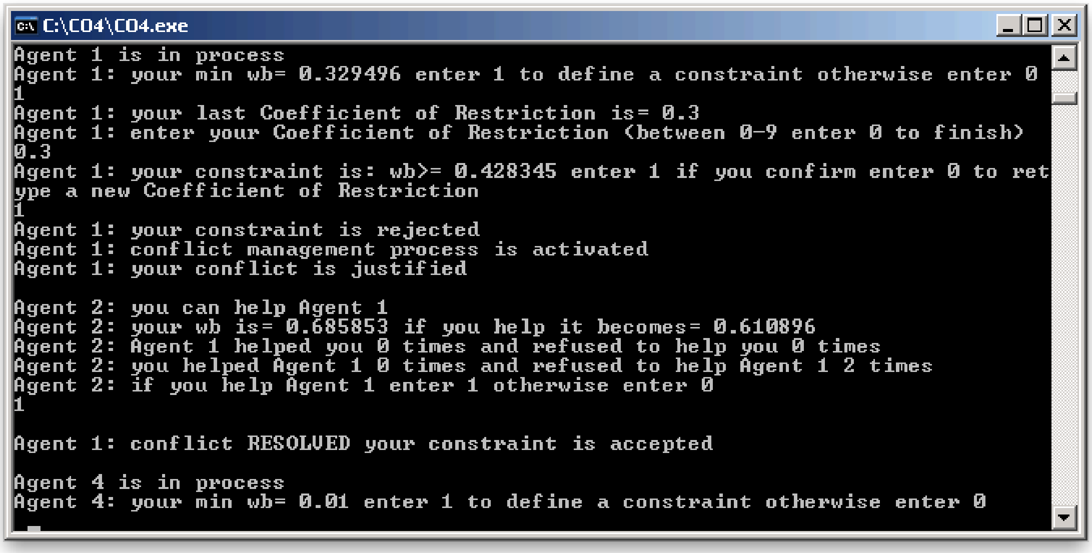
If there is at least one feasible help, the most optimal one is detected. In Strategy 4, the conflict is resolved directly without any approval request from the helping agent. As shown in Figure 5.6, the helping agent is informed when the conflict is resolved.



```
Agent 1: your last Coefficient of Restriction is= 9
Agent 1: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
3
Agent 1: your constraint is: wb>= 0.4 enter 1 if you confirm enter 0 to retype a
 new Coefficient of Restriction
1
Agent 1: your constraint is rejected
Agent 1: conflict management process is activated
Agent 1: your conflict is justified

Agent 2: you help Agent 1

Agent 1: conflict RESOLVED your constraint is accepted


Iteration: 9
Agent 1: WB= 0.4
Agent 2: WB= 0.59305
Agent 3: WB= 0.01
Agent 4: WB= 1

Agent 2 is in process
Agent 2: your min wb= 0.59305 enter 1 to define a constraint otherwise enter 0
```

Figure 5.6: Report of conflict resolution in Strategy 4

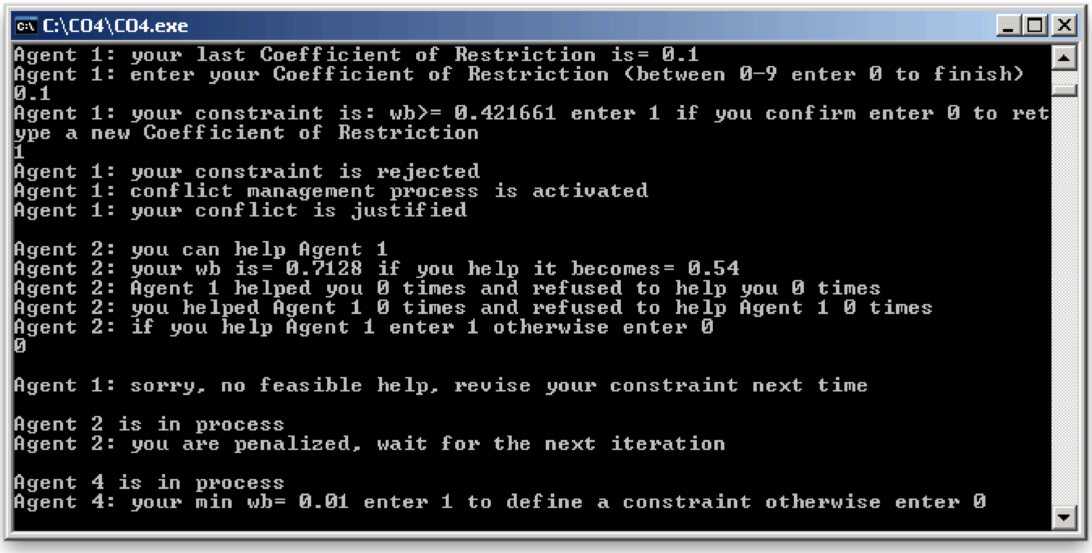
In Strategies 2 and 3, the approval of the helping agent is requested. As shown in Figure 5.7, the agent that can help is informed about what his/her wellbeing value will become if help is approved, and how many times agents have helped each other in previous iterations. If help is approved, then the conflict is resolved.



```
Agent 1 is in process
Agent 1: your min wb= 0.329496 enter 1 to define a constraint otherwise enter 0
1
Agent 1: your last Coefficient of Restriction is= 0.3
Agent 1: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
0.3
Agent 1: your constraint is: wb>= 0.428345 enter 1 if you confirm enter 0 to ret
ype a new Coefficient of Restriction
1
Agent 1: your constraint is rejected
Agent 1: conflict management process is activated
Agent 1: your conflict is justified

Agent 2: you can help Agent 1
Agent 2: your wb is= 0.685853 if you help it becomes= 0.610896
Agent 2: Agent 1 helped you 0 times and refused to help you 0 times
Agent 2: you helped Agent 1 0 times and refused to help Agent 1 2 times
Agent 2: if you help Agent 1 enter 1 otherwise enter 0
1

Agent 1: conflict RESOLVED your constraint is accepted

Agent 4 is in process
Agent 4: your min wb= 0.01 enter 1 to define a constraint otherwise enter 0
```

Figure 5.7: Report of feasible help in Strategies 2 and 3

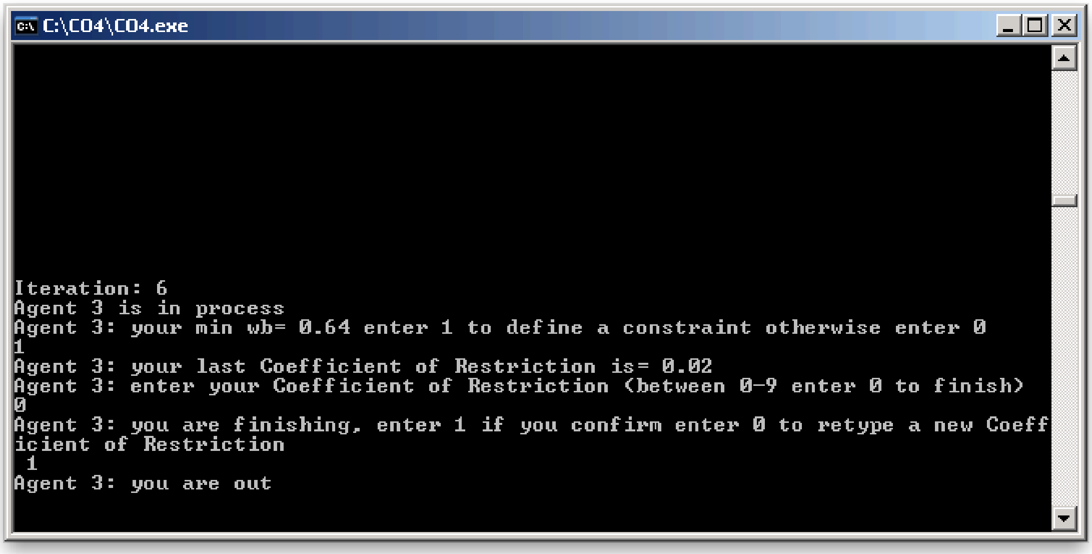
If help is not approved, the agent should revise the constraint at next iteration. In Strategy 3, the agent that refuses to help is penalized. He/she cannot define a constraint at next iteration. This is reported to the agent as shown in Figure 5.8.



```
Agent 1: your last Coefficient of Restriction is= 0.1
Agent 1: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
0.1
Agent 1: your constraint is: wb>= 0.421661 enter 1 if you confirm enter 0 to ret
ype a new Coefficient of Restriction
1
Agent 1: your constraint is rejected
Agent 1: conflict management process is activated
Agent 1: your conflict is justified

Agent 2: you can help Agent 1
Agent 2: your wb is= 0.7128 if you help it becomes= 0.54
Agent 2: Agent 1 helped you 0 times and refused to help you 0 times
Agent 2: you helped Agent 1 0 times and refused to help Agent 1 0 times
Agent 2: if you help Agent 1 enter 1 otherwise enter 0
0

Agent 1: sorry, no feasible help, revise your constraint next time

Agent 2 is in process
Agent 2: you are penalized, wait for the next iteration

Agent 4 is in process
Agent 4: your min wb= 0.01 enter 1 to define a constraint otherwise enter 0
```

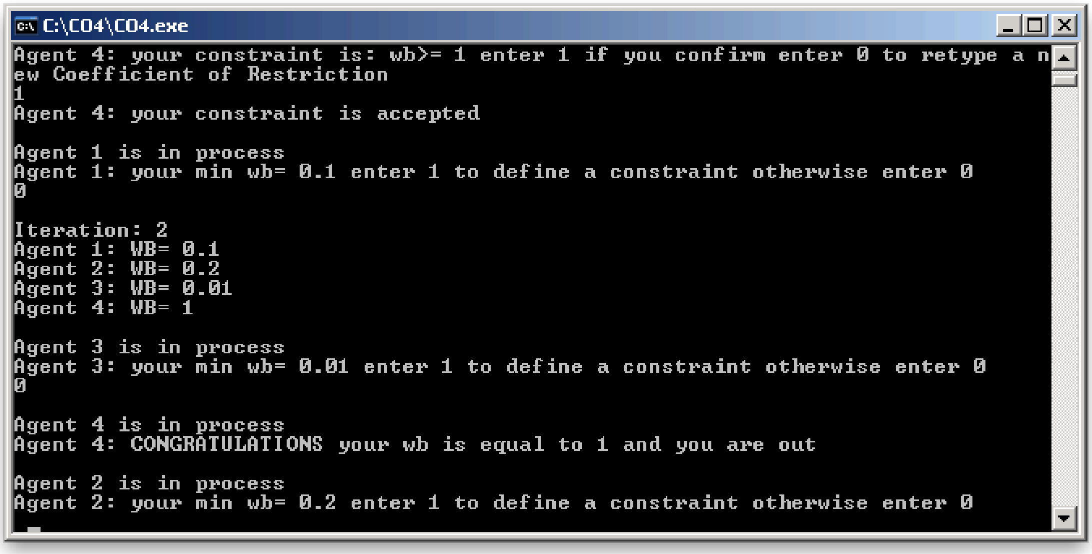
Figure 5.8: Report of penalization in Strategy 3

Agents can finish their part in the game by two ways: intentionally -when they enter 0 as the coefficient of restriction (shown in Figure 5.9) - , or automatically - when their minimum wellbeing is equal to 1 (shown in Figure 5.10) -. If an agent is out he/she cannot define constraints anymore. However, he/she can still receive messages if another agent in process requires his/her help. Thus, agents need to follow their screen until the game is finished. If the agent is out and approves help; since his/her wellbeing is reduced, he/she can reenter to the game. The game is finished when all the four agents are out.



```
Iteration: 6
Agent 3 is in process
Agent 3: your min wb= 0.64 enter 1 to define a constraint otherwise enter 0
1
Agent 3: your last Coefficient of Restriction is= 0.02
Agent 3: enter your Coefficient of Restriction (between 0-9 enter 0 to finish)
0
Agent 3: you are finishing, enter 1 if you confirm enter 0 to retype a new Coeff
icient of Restriction
 1
Agent 3: you are out
```

Figure 5.9: Finishing the game intentionally



```
Agent 4: your constraint is: wb>= 1 enter 1 if you confirm enter 0 to retype a n
ew Coefficient of Restriction
1
Agent 4: your constraint is accepted

Agent 1 is in process
Agent 1: your min wb= 0.1 enter 1 to define a constraint otherwise enter 0
0

Iteration: 2
Agent 1: WB= 0.1
Agent 2: WB= 0.2
Agent 3: WB= 0.01
Agent 4: WB= 1

Agent 3 is in process
Agent 3: your min wb= 0.01 enter 1 to define a constraint otherwise enter 0
0

Agent 4 is in process
Agent 4: CONGRATULATIONS your wb is equal to 1 and you are out

Agent 2 is in process
Agent 2: your min wb= 0.2 enter 1 to define a constraint otherwise enter 0
```

Figure 5.10: Finishing the game automatically

5.3 Experiment Results

Eight members of Ecole Centrale Paris Industrial Engineering Laboratory participated to the experiments. Experiments were thus conducted with two different groups of people (Group 1 and Group2). In the ideal situation, experiments should be conducted with various groups, so that results show a statistical significance. However, each experiment takes around two and a half hours. Considering the limited time, we opted for experimenting with only two groups. The results are however comparable when the hypothesis “a human agent is an entity that reflects always the same character” is accepted.

Strategies are experimented sequentially. Each strategy is conducted as a separate game. Initial wellbeing values of agents are equal to 0.1 in each game. The minimum wellbeing of an agent can have three different values at process iteration:

- **Beginning value:** the minimum wellbeing value that the agent has at the beginning of iteration
- **Intermediate value:** the minimum wellbeing value that the agent has just after defining his/her constraint
- **Ending value:** the minimum wellbeing value that the agent has at the end of iteration

If the constraint of an agent is rejected, the intermediate value of his/her minimum wellbeing is equal to its beginning value. The ending value of an agent’s minimum wellbeing can be different than its intermediate value if the agent helps another agent. Otherwise they are equal. Since there is no conflict resolution in Strategy 1, intermediate and ending values are equal. If the intermediate value is less than the constraint value V where $wb \geq V$ at an iteration, it represents a potential conflict. It means that the constraint is rejected. The constraint value V is 0 either when the agent does not define a constraint (on purpose to leave space to the other agents, or when the agent is out of the game), or is punished in Strategy 4. The evolution of these values in different strategies for both Group 1 and Group 2 are shown in following subsections. Number of iterations, total satisfaction, and satisfaction divergence values of four strategies are also compared for both Group 1 and Group 2.

5.3.1 Results of Group 1

Agents in Group 1 enter their compromise threshold values as shown in Table 5.1.

Table 5.1: Compromise threshold values of agents in Group 1

	Compromise Threshold
Agent 1:	0.7
Agent 2:	0.6
Agent 3:	0.8
Agent 4:	0.9

5.3.1.1 Strategy 1

The results of Strategy 1 with Group 1 are shown in Figures 5.11-5.14.

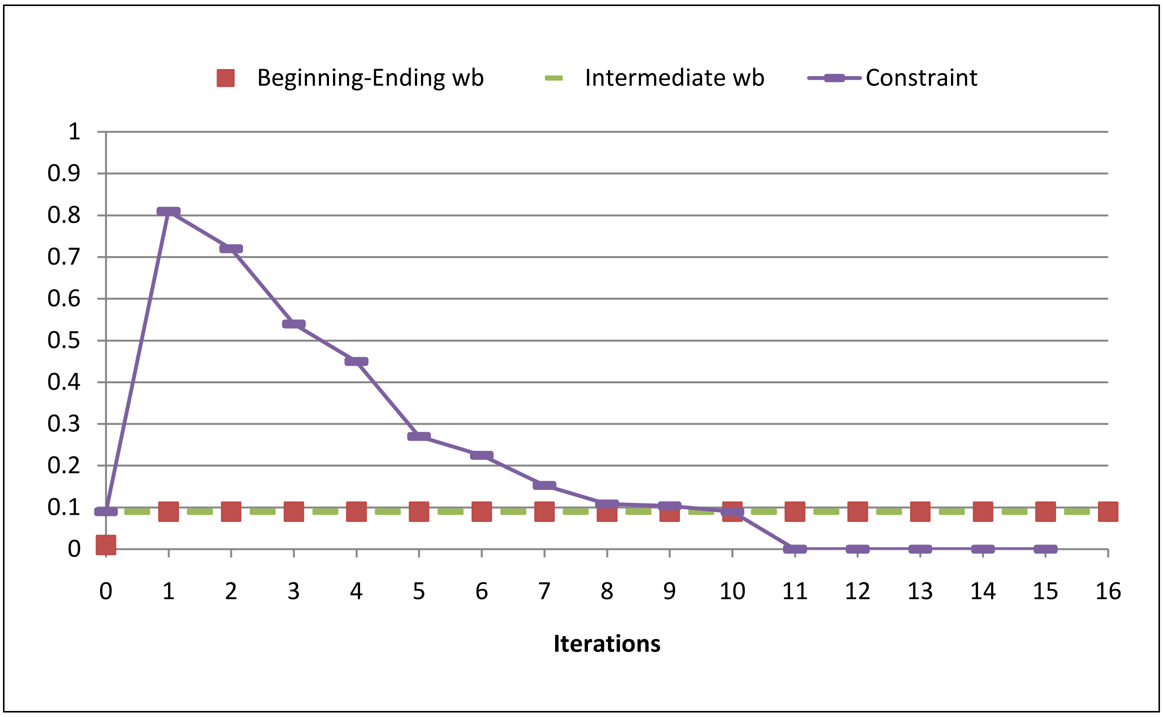


Figure 5.11: Results of Agent 1 (Group 1) in Strategy 1

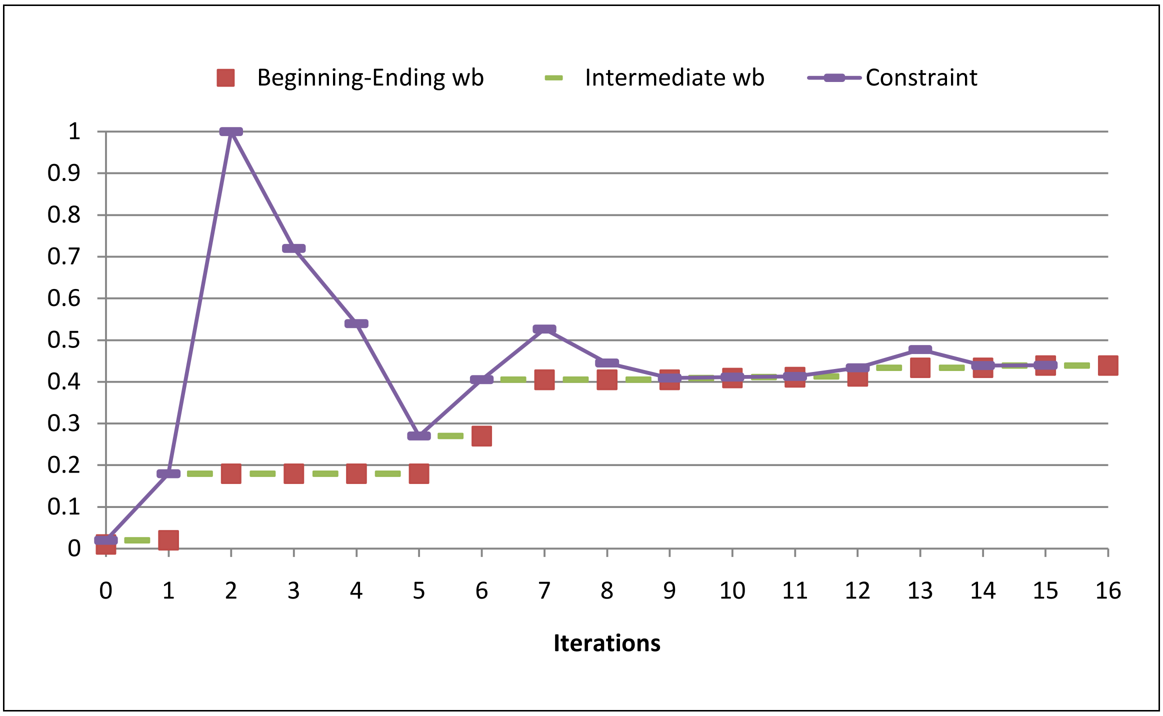


Figure 5.12: Results of Agent 2 (Group 1) in Strategy 1

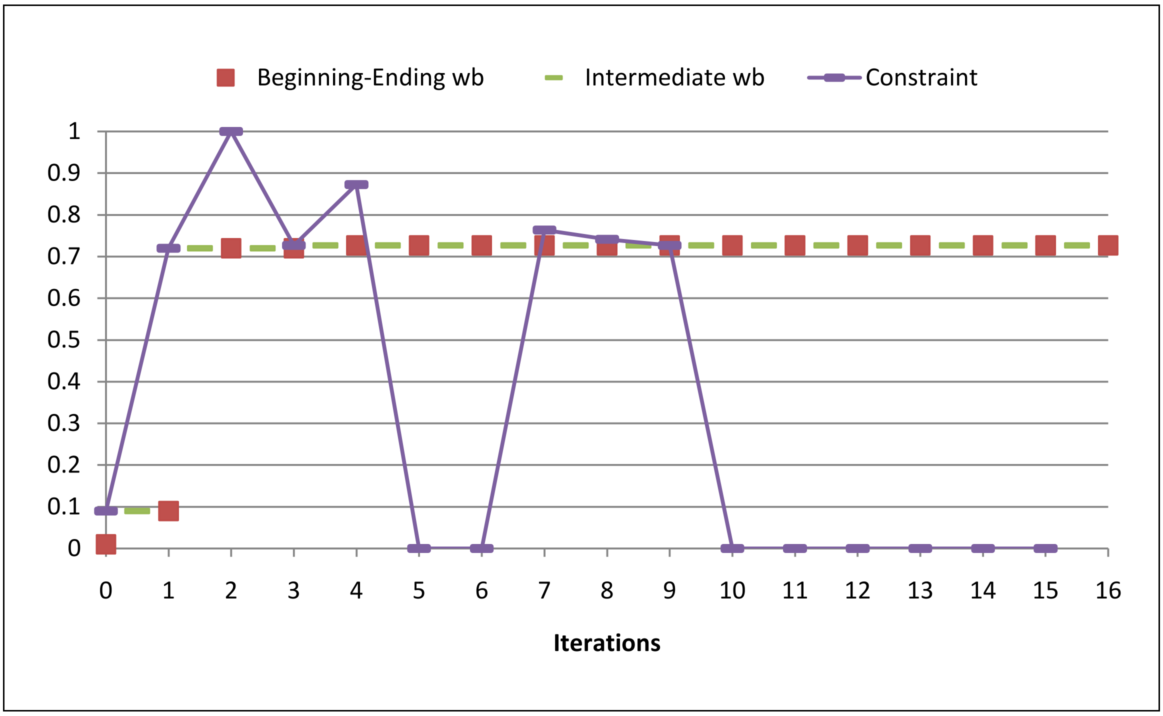


Figure 5.13: Results of Agent 3 (Group 1) in Strategy 1

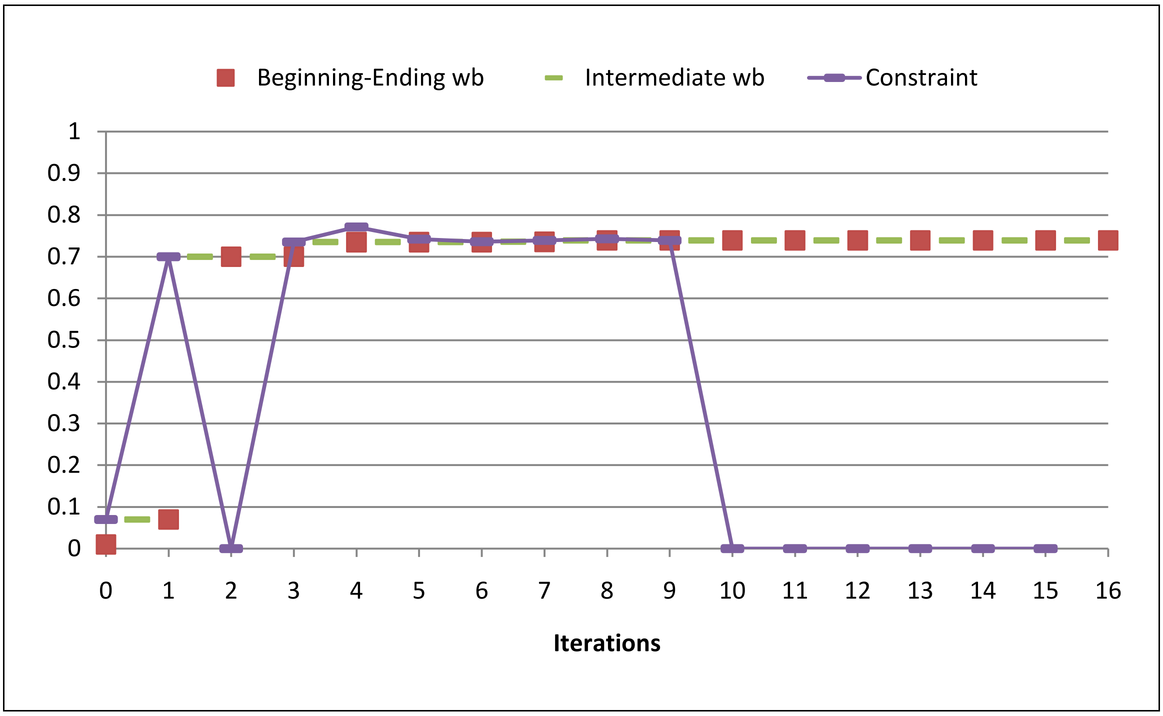


Figure 5.14: Results of Agent 4 (Group 1) in Strategy 1

5.3.1.2 Strategy 2

The results of Strategy 2 with Group 1 are shown in Figures 5.15-5.18.

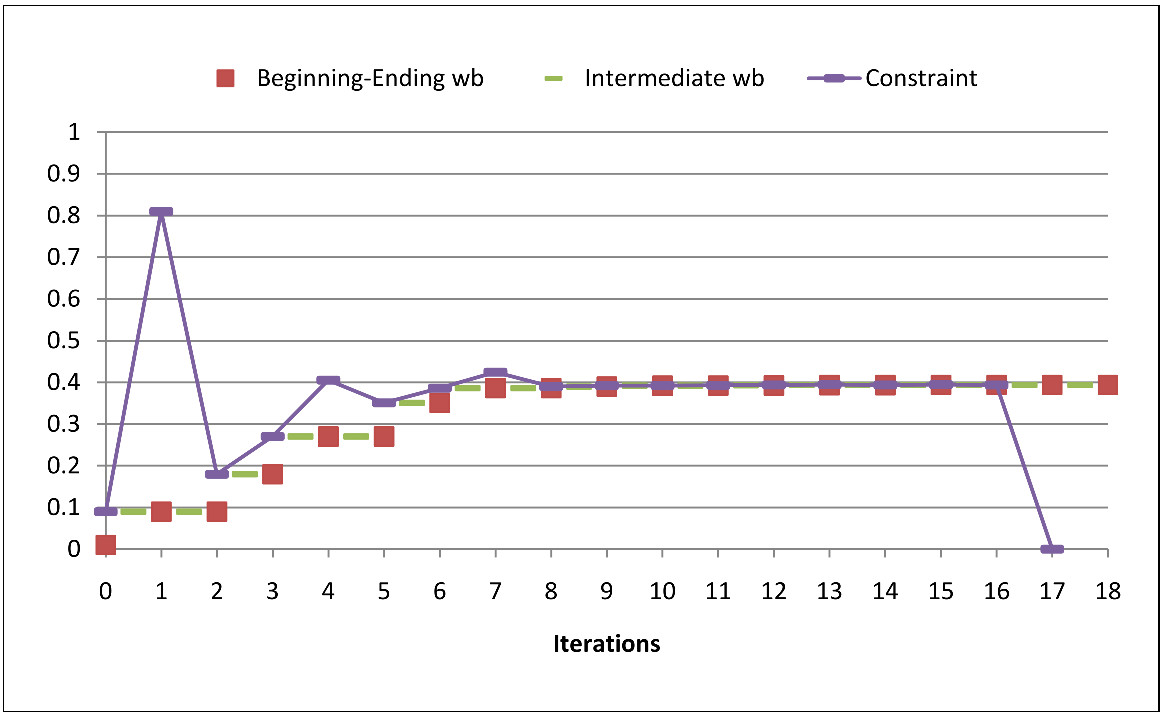


Figure 5.15: Results of Agent 1 (Group 1) in Strategy 2

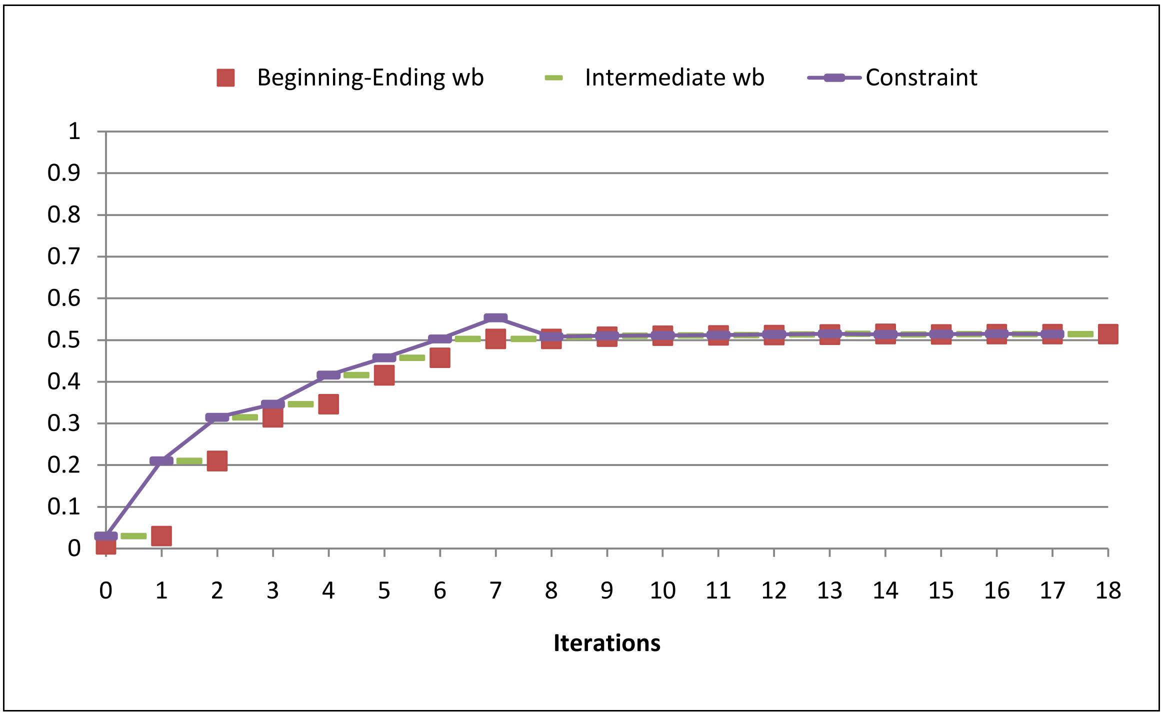


Figure 5.16: Results of Agent 2 (Group 1) in Strategy 2

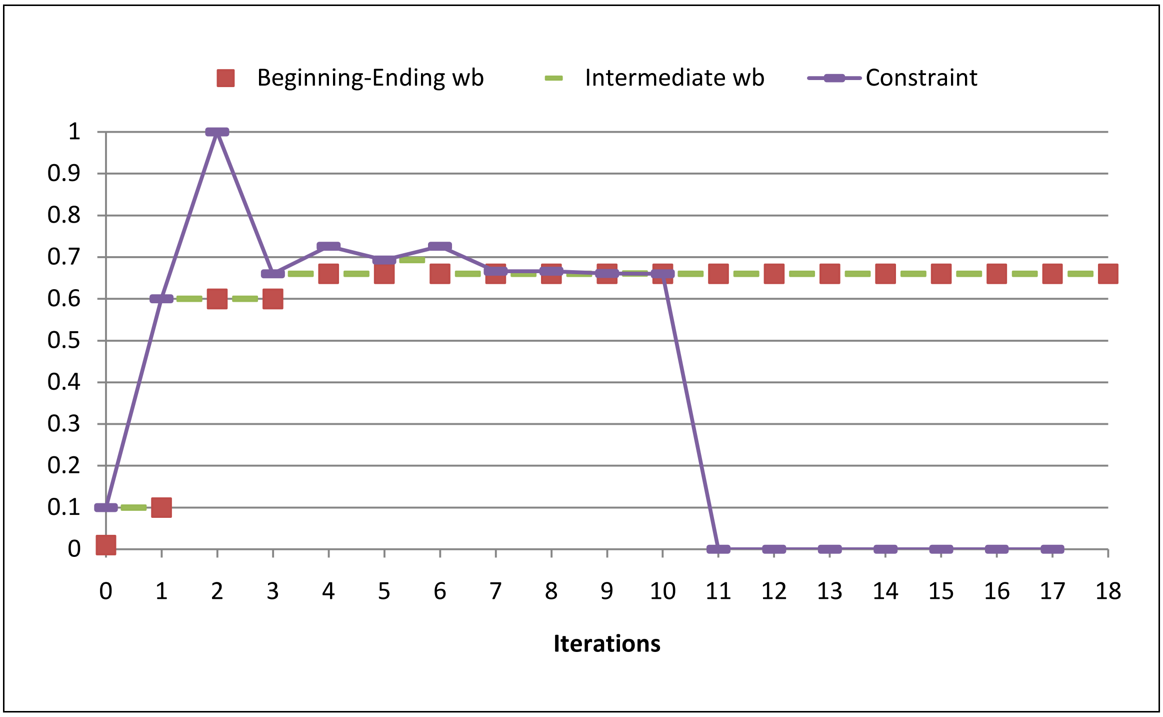


Figure 5.17: Results of Agent 3 (Group 1) in Strategy 2

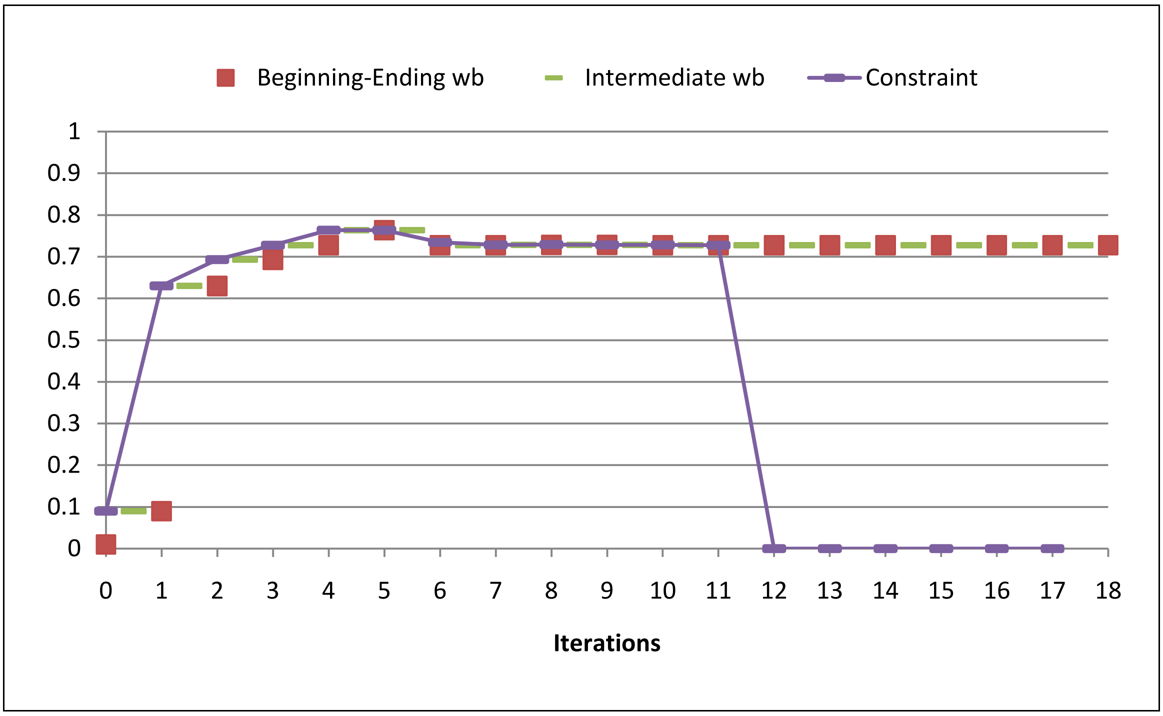


Figure 5.18: Results of Agent 4 (Group 1) in Strategy 2

5.3.1.3 Strategy 3

The results of Strategy 3 with Group 1 are shown in Figures 5.19-5.22.

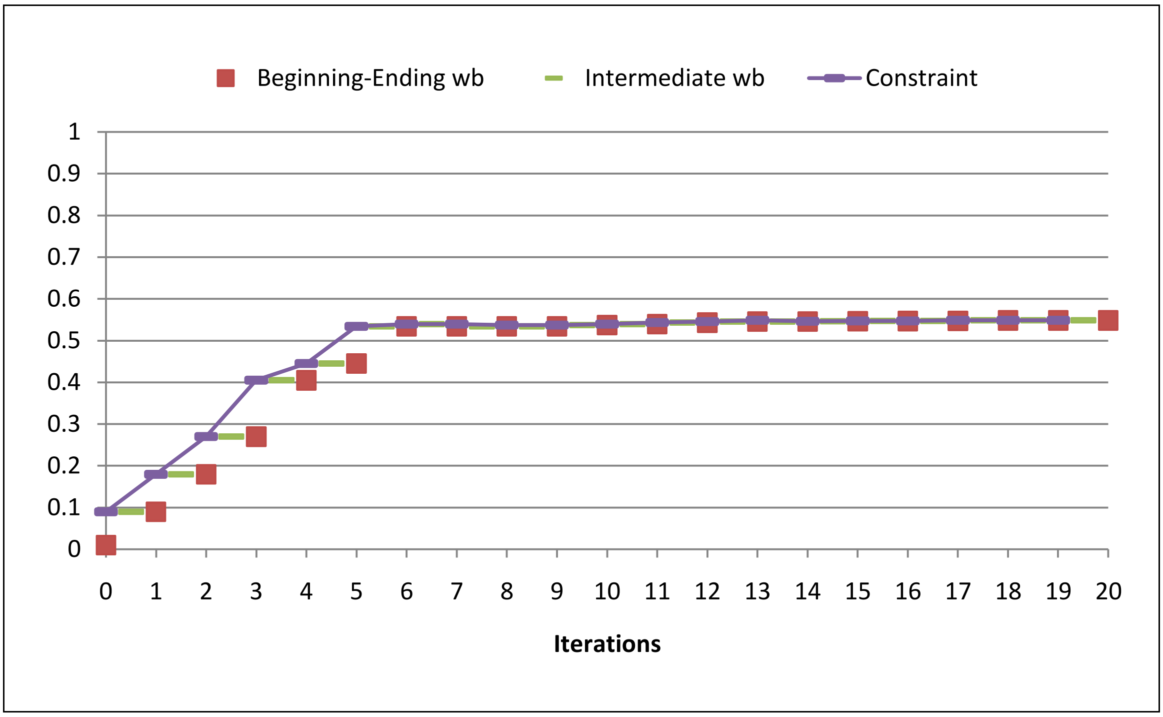


Figure 5.19: Results of Agent 1 (Group 1) in Strategy 3

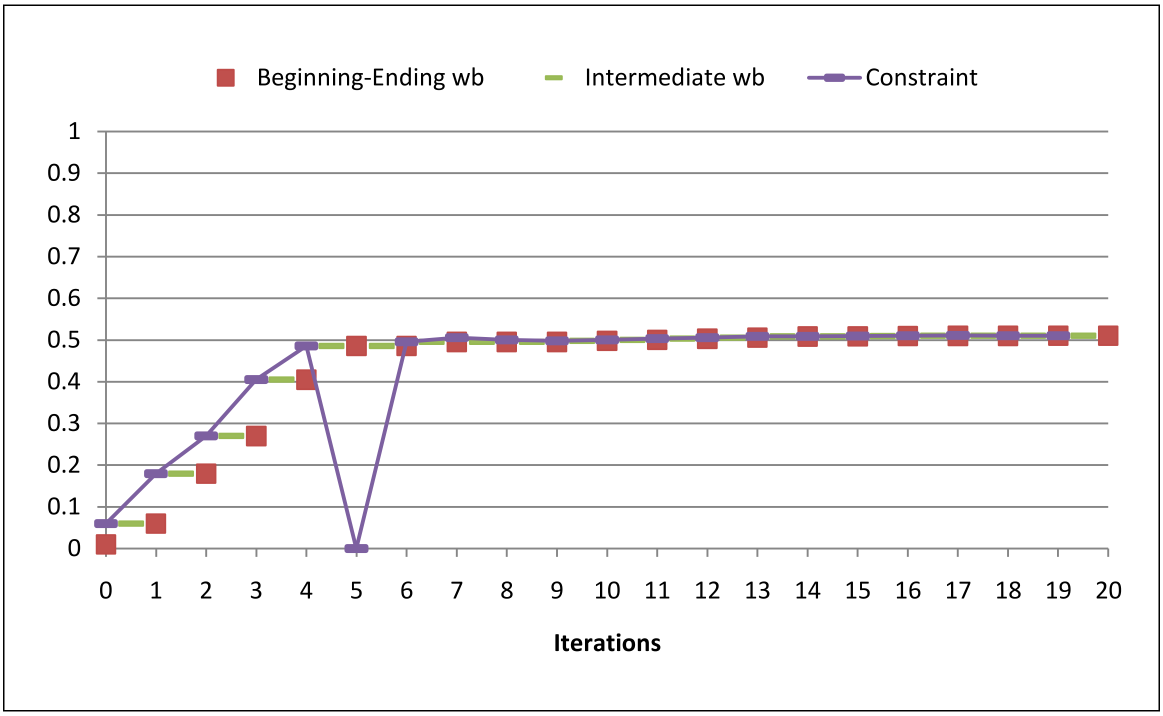


Figure 5.20: Results of Agent 2 (Group 1) in Strategy 3

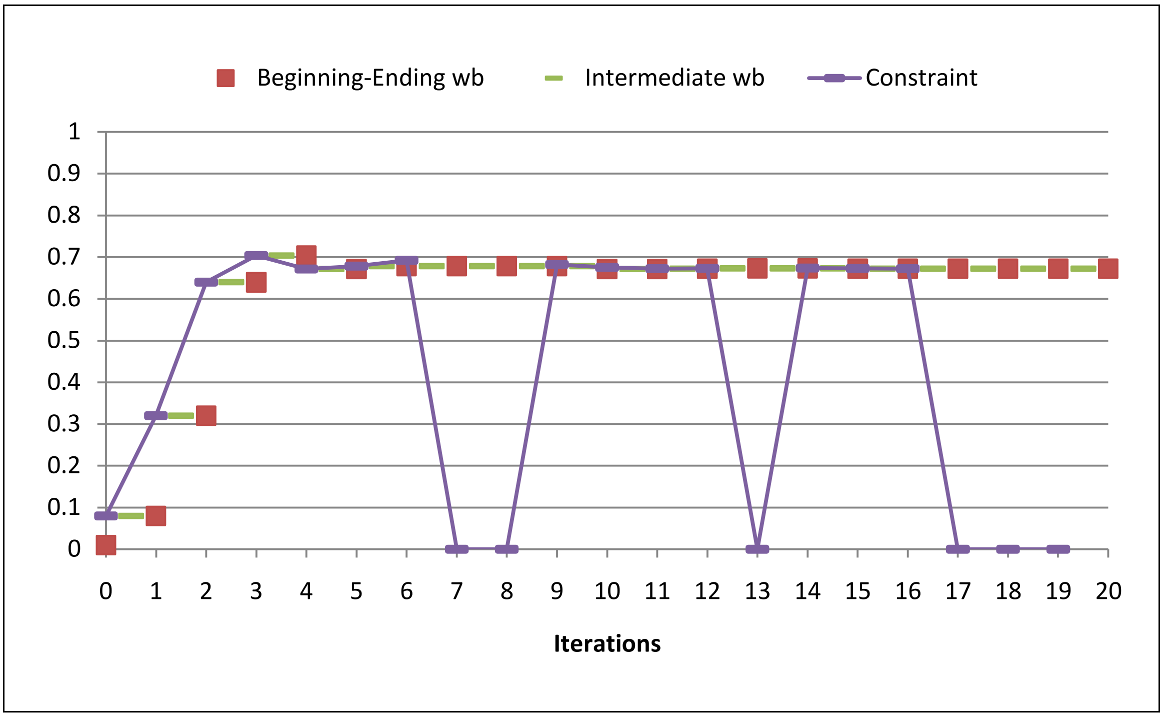


Figure 5.21: Results of Agent 3 (Group 1) in Strategy 3

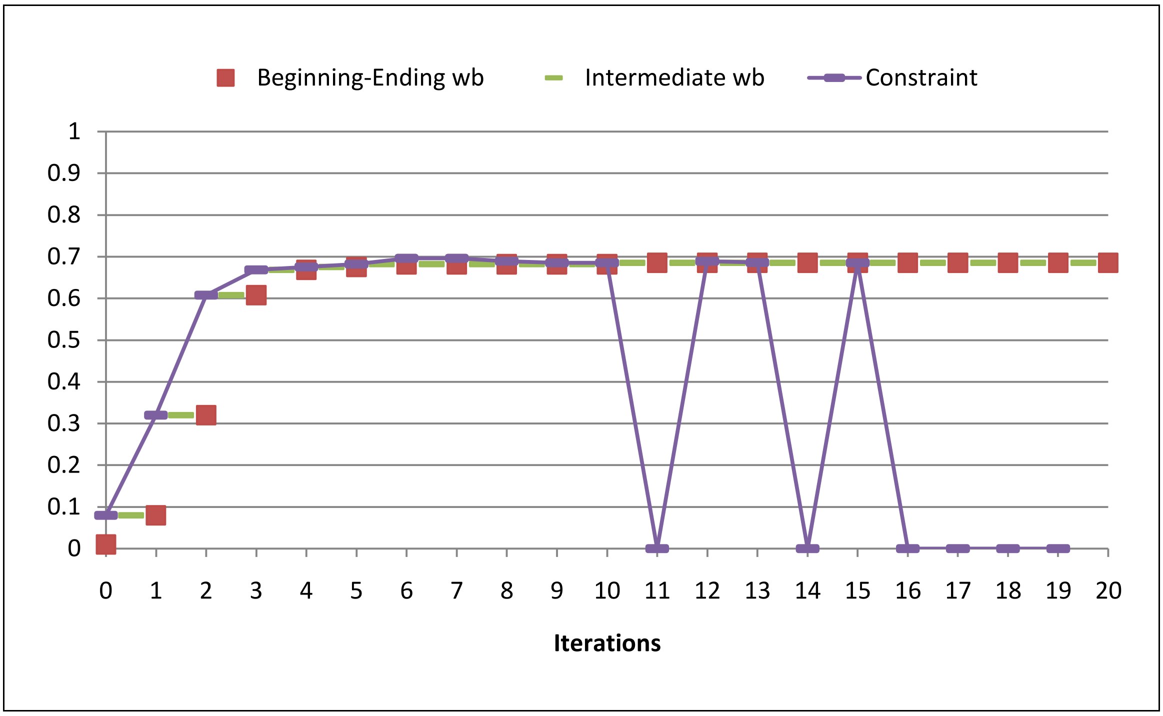


Figure 5.22: Results of Agent 4 (Group 1) in Strategy 3

5.3.1.4 Strategy 4

The results of Strategy 4 with Group 1 are shown in Figures 5.23-5.26.

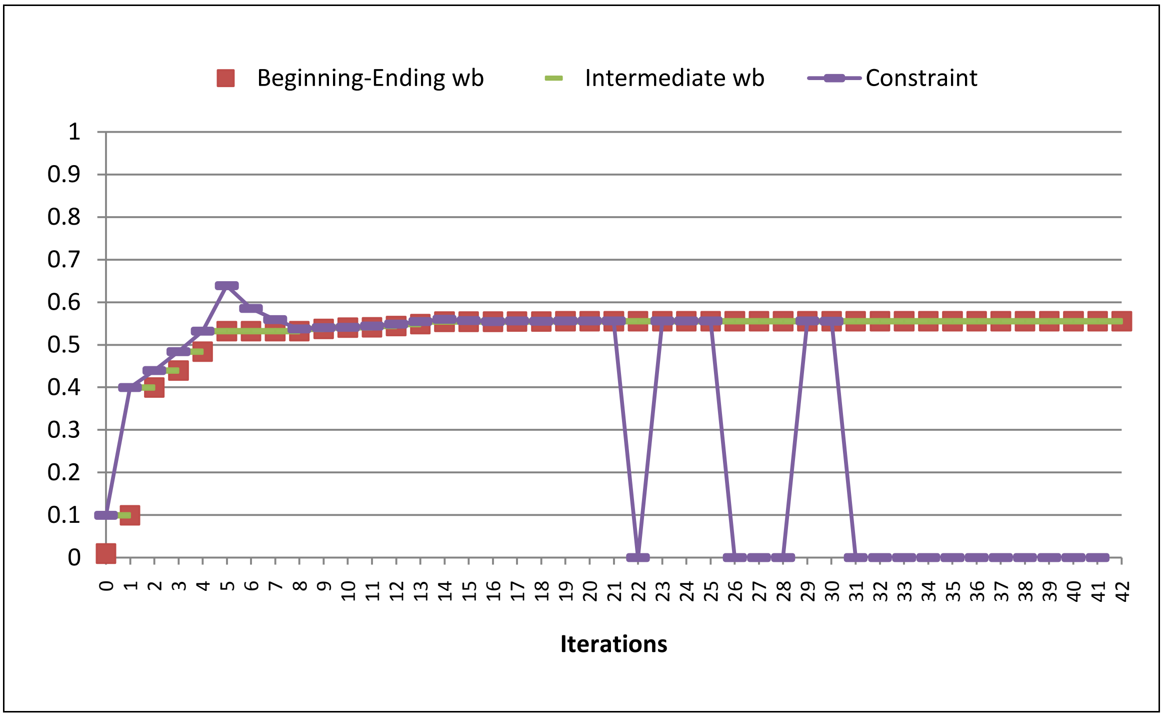


Figure 5.23: Results of Agent 1 (Group 1) in Strategy 4

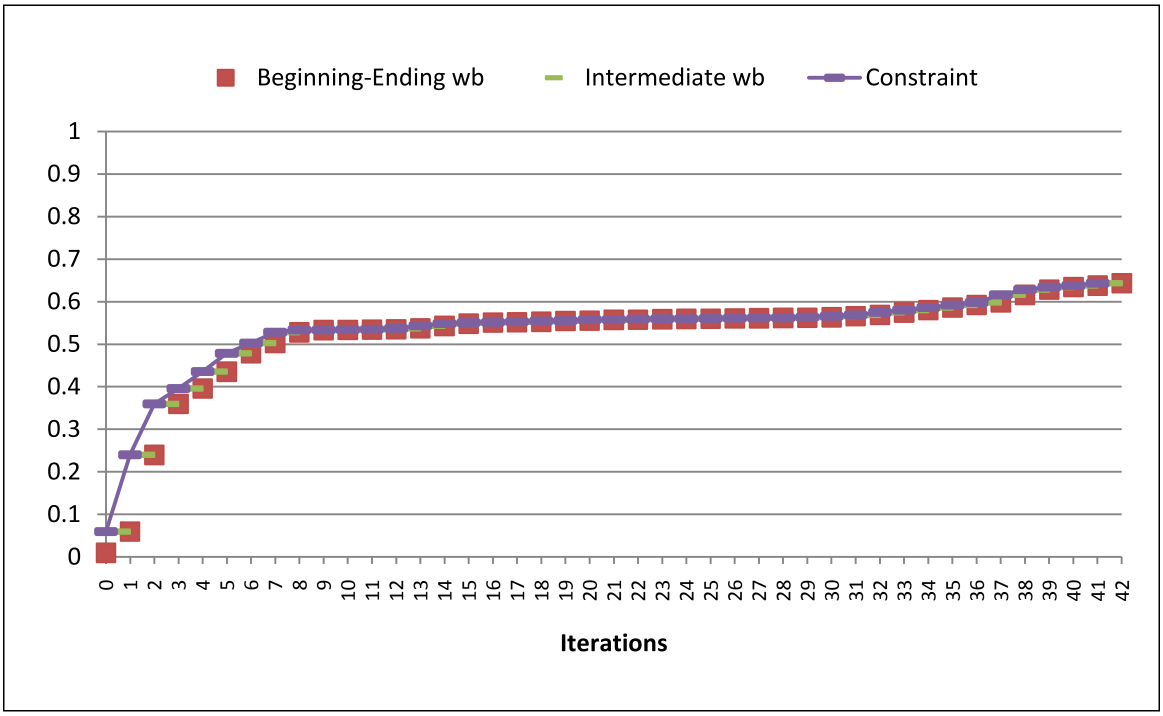


Figure 5.24: Results of Agent 2 (Group 1) in Strategy 4

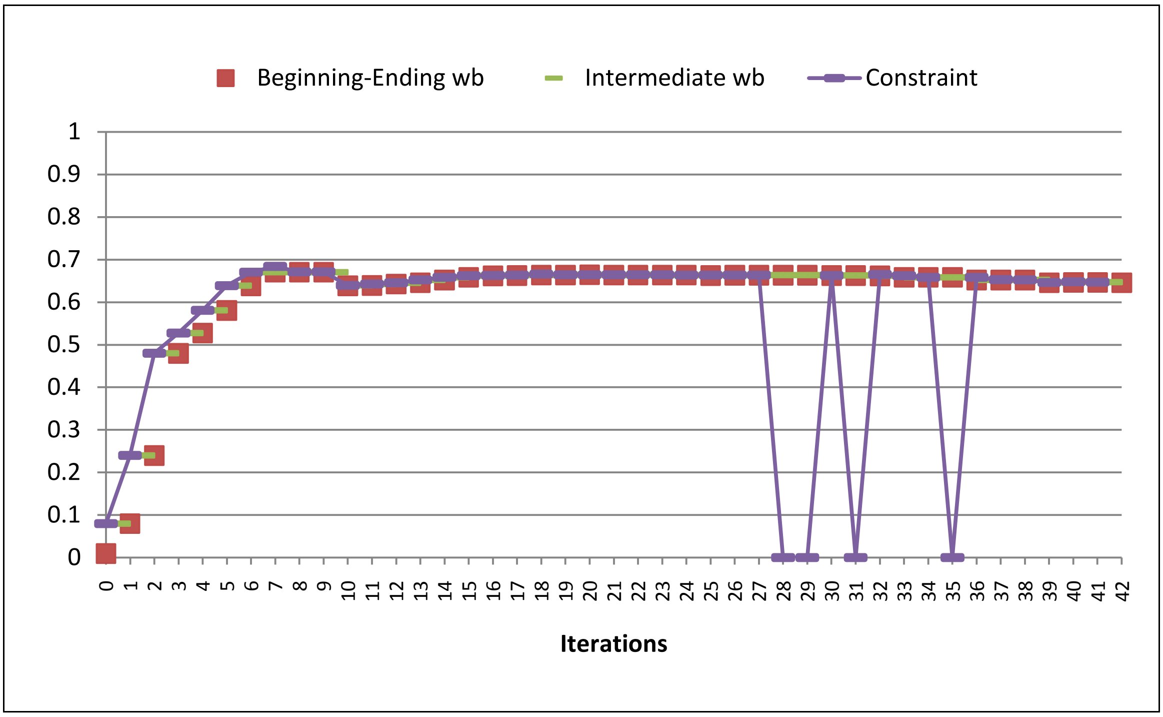


Figure 5.25: Results of Agent 3 (Group 1) in Strategy 4

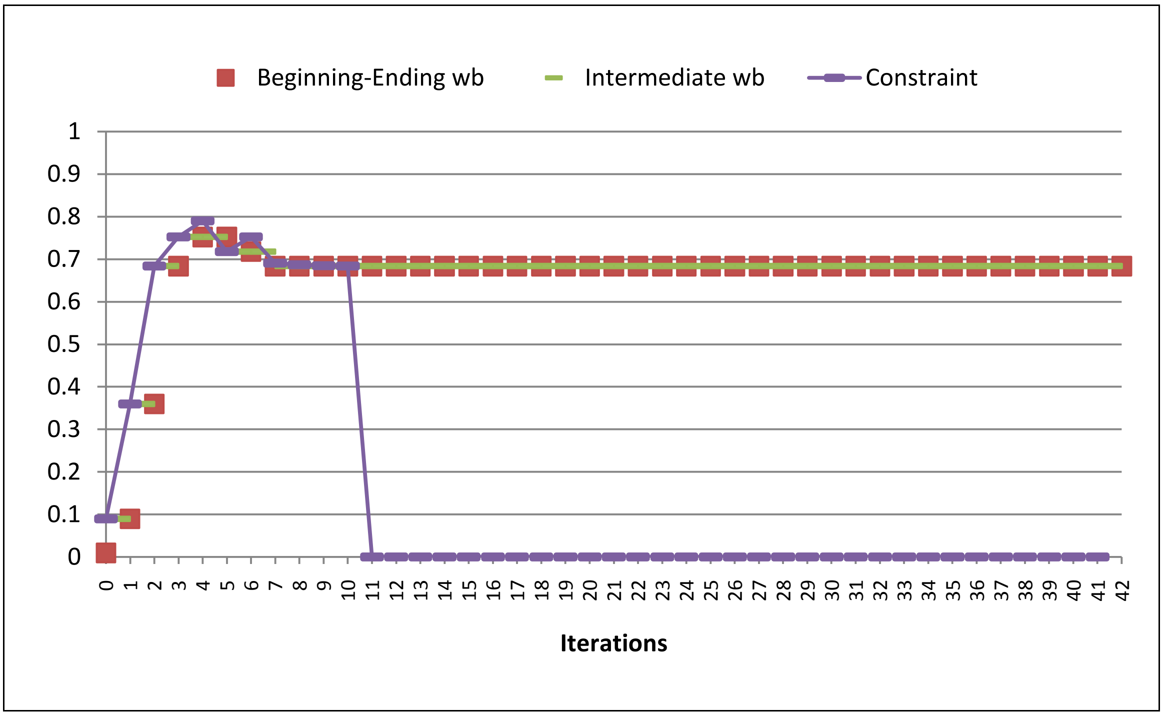


Figure 5.26: Results of Agent 4 (Group 1) in Strategy 4

5.3.1.5 Comparisons

As seen in Figure 5.11, in Strategy 1, Agent 1 could not improve his/her wellbeing, because he/she was blocked by the modifications of the other agents. Also, Agent 2 was not very successful in this strategy. Their unsuccessful attempts can be seen as large differences between some constraint values and intermediate wellbeing values in Figure 5.11 and 5.12. Their attempts were more successful in Strategies 2, 3 and 4, because of the conflict resolutions. Conflict resolutions can be noticed by the reduction of minimum wellbeing values in Figures 5.11-5.26. This reduction allows the other agent in conflict to improve his/her wellbeing. Some of these reductions are more obvious where the wellbeing value is reduced explicitly (e.g. Agent 4 at iterations 5-6 in Strategy 4); while some of the resolutions are less obvious. Total number of resolved conflicts: 5 in Strategy 2, 4 in Strategy 3, and 13 in Strategy 4.

The evolutions of the total satisfaction and satisfaction divergence values are compared in Figures 5.27 and 5.28. Around 90% of the final total satisfaction is obtained at first four iterations. This does not mean that the process should have been terminated at this level, because at further iterations the satisfaction divergence could be reduced by resolving conflicts while maintaining or even improving the total satisfaction. Strategy

4 resulted in the largest final total satisfaction (1.908), followed by Strategy 3 (1.846), Strategy 2 (1.767), and Strategy 1 (1.574). Strategy 4 resulted in the smallest final satisfaction divergence (0.383), followed by Strategy 3 (0.49), Strategy 2 (0.63), and Strategy 1 (0.97).

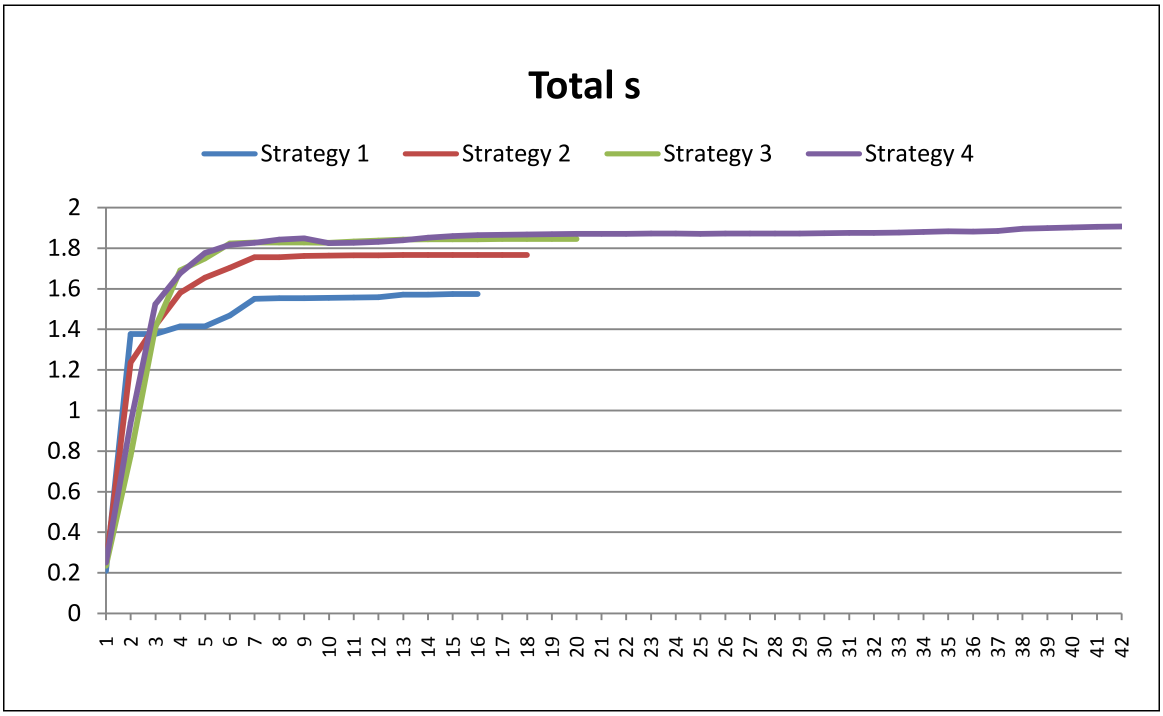


Figure 5.27: Comparison of the total satisfaction values (Group 1)

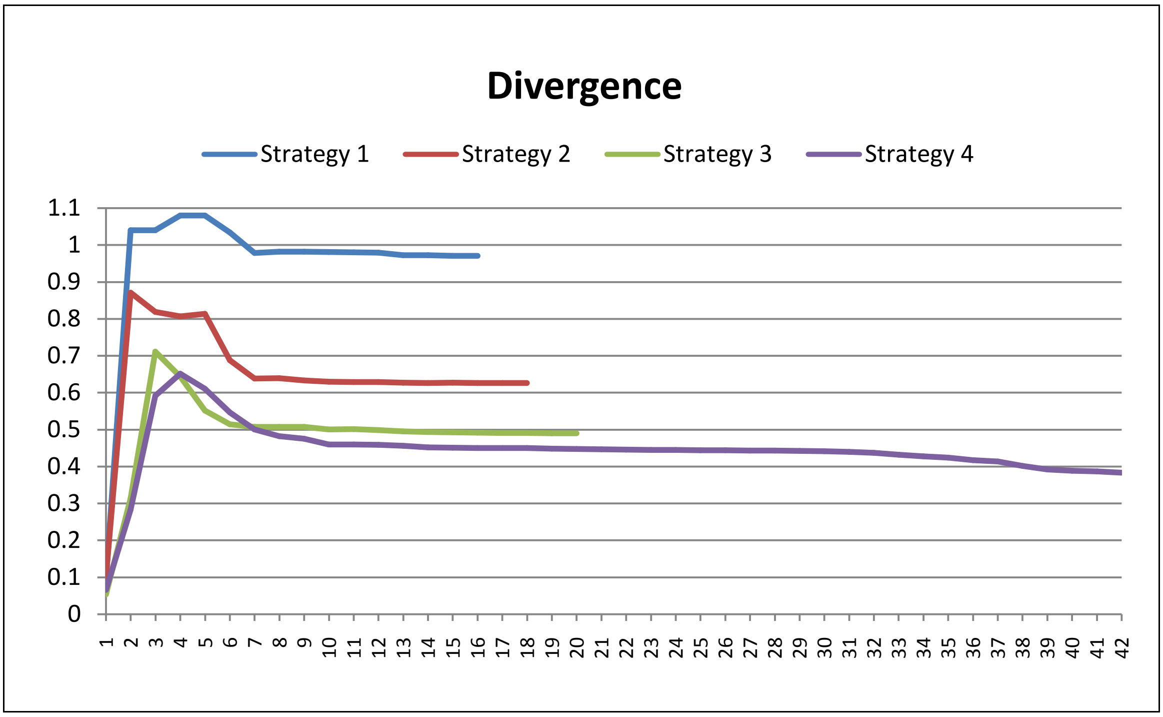


Figure 5.28: Comparison of the satisfaction divergence values (Group 1)

5.3.2 Results of Group 2

Agents in Group 2 enter their compromise threshold values as shown in Table 5.2. They all defined egoistic threshold values.

Table 5.2: Compromise threshold values of agents in Group 2

	Compromise Threshold
Agent 1:	1
Agent 2:	1
Agent 3:	1
Agent 4:	1

5.3.2.1 Strategy 1

The results of Strategy 1 with Group 2 are shown in Figures 5.29-5.32.

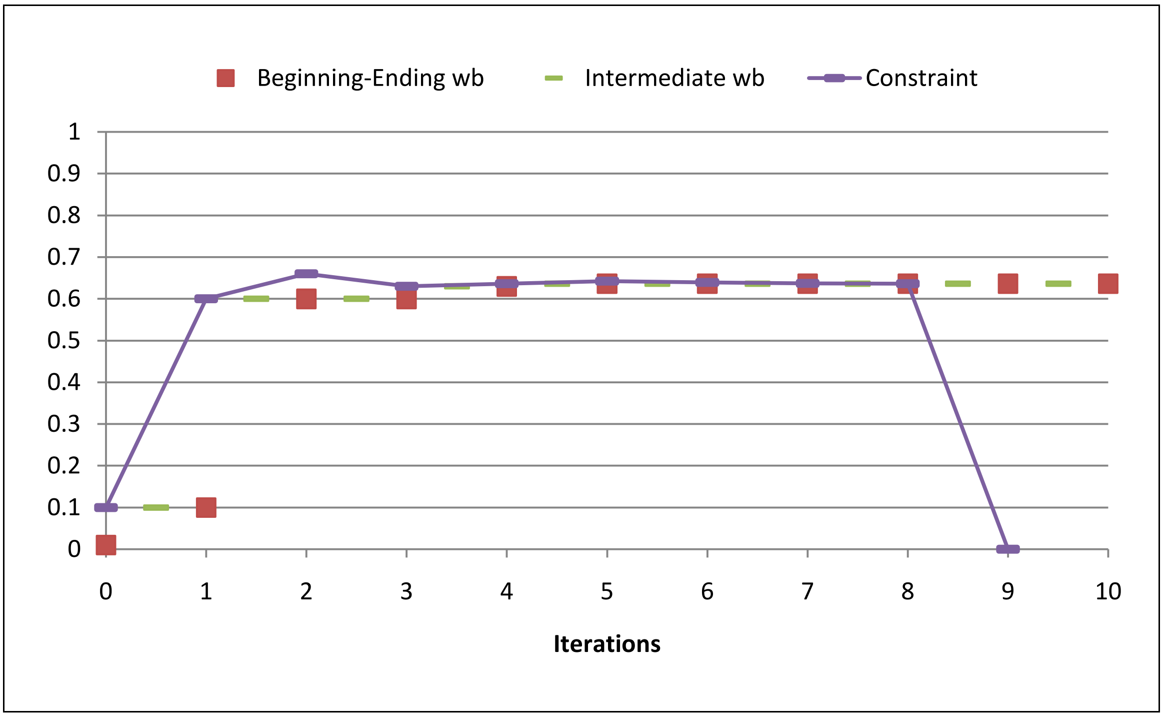


Figure 5.29: Results of Agent 1 (Group 2) in Strategy 1

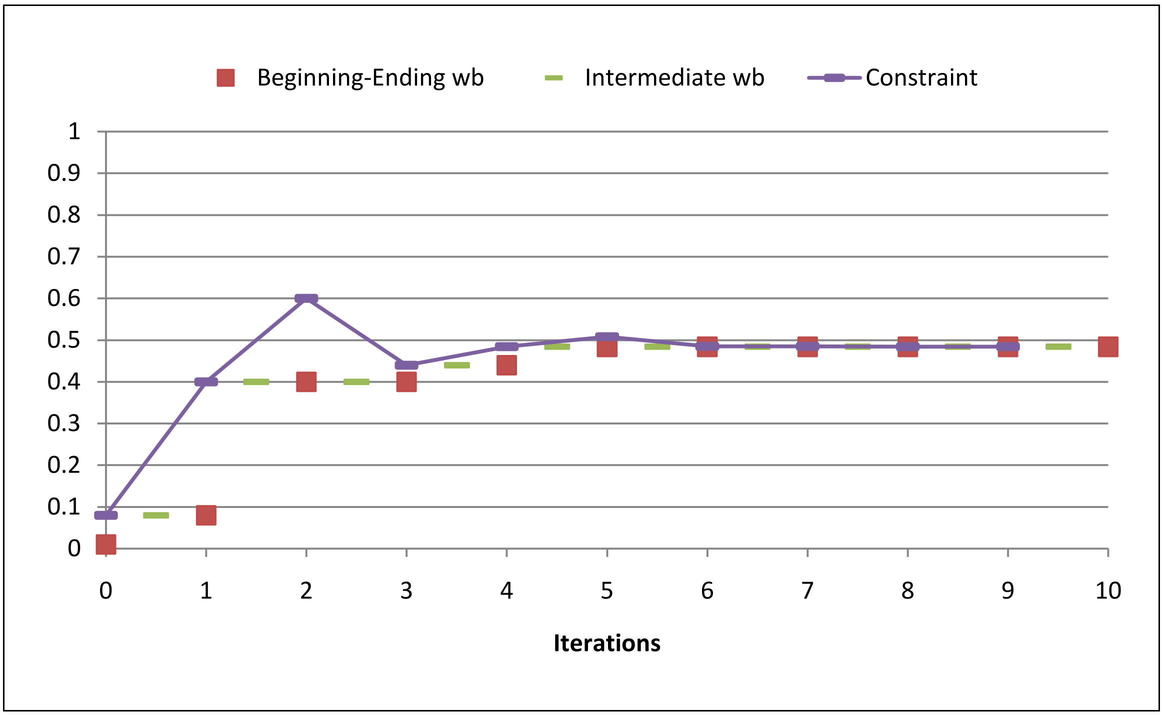


Figure 5.30: Results of Agent 2 (Group 2) in Strategy 1

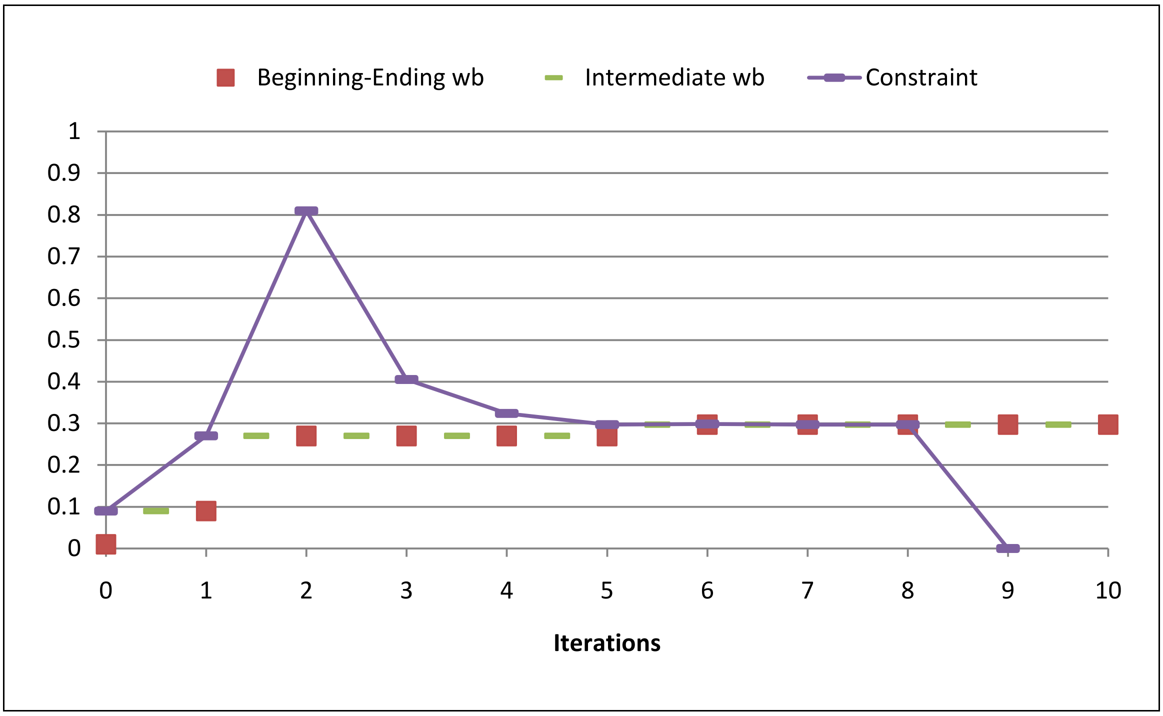


Figure 5.31: Results of Agent 3 (Group 2) in Strategy 1

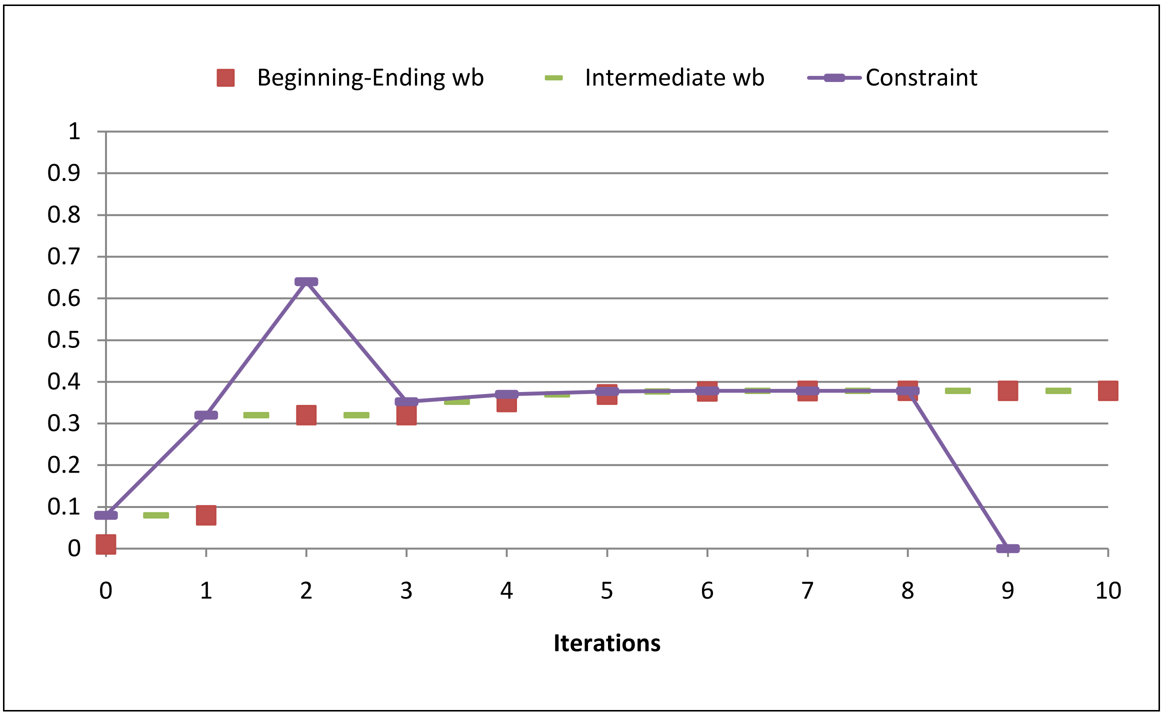


Figure 5.32: Results of Agent 4 (Group 2) in Strategy 1

5.3.2.2 Strategy 2

The results of Strategy 2 with Group 2 are shown in Figures 5.33-5.36.

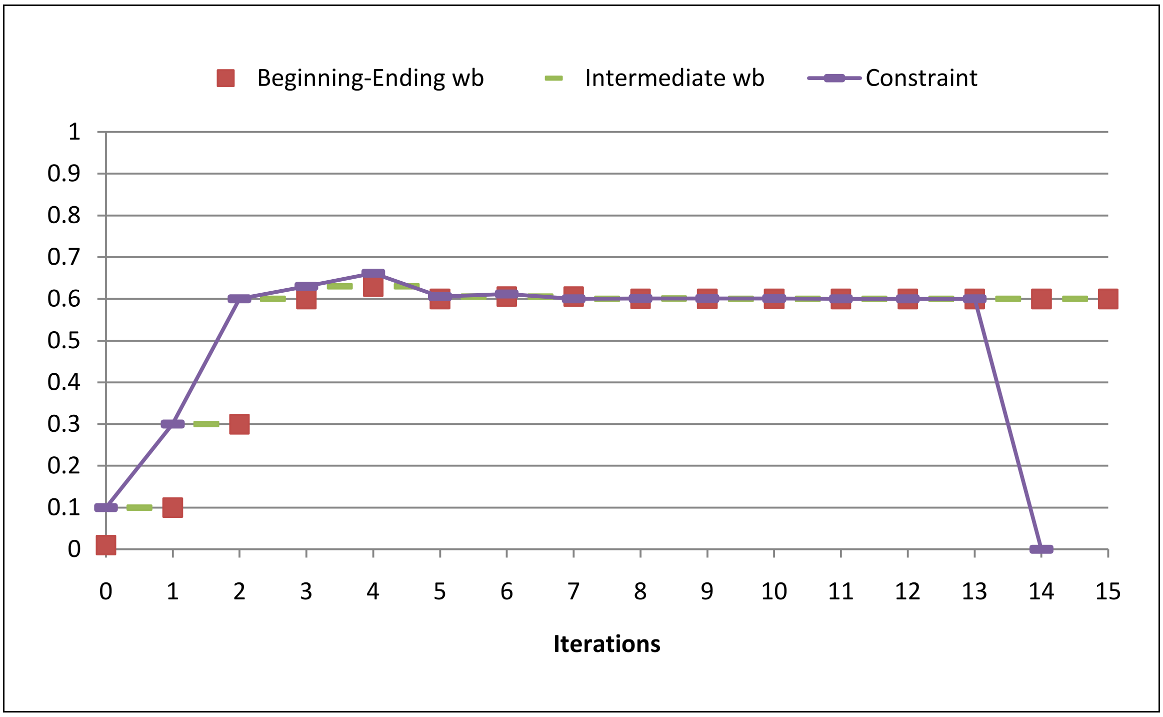


Figure 5.33: Results of Agent 1 (Group 2) in Strategy 2

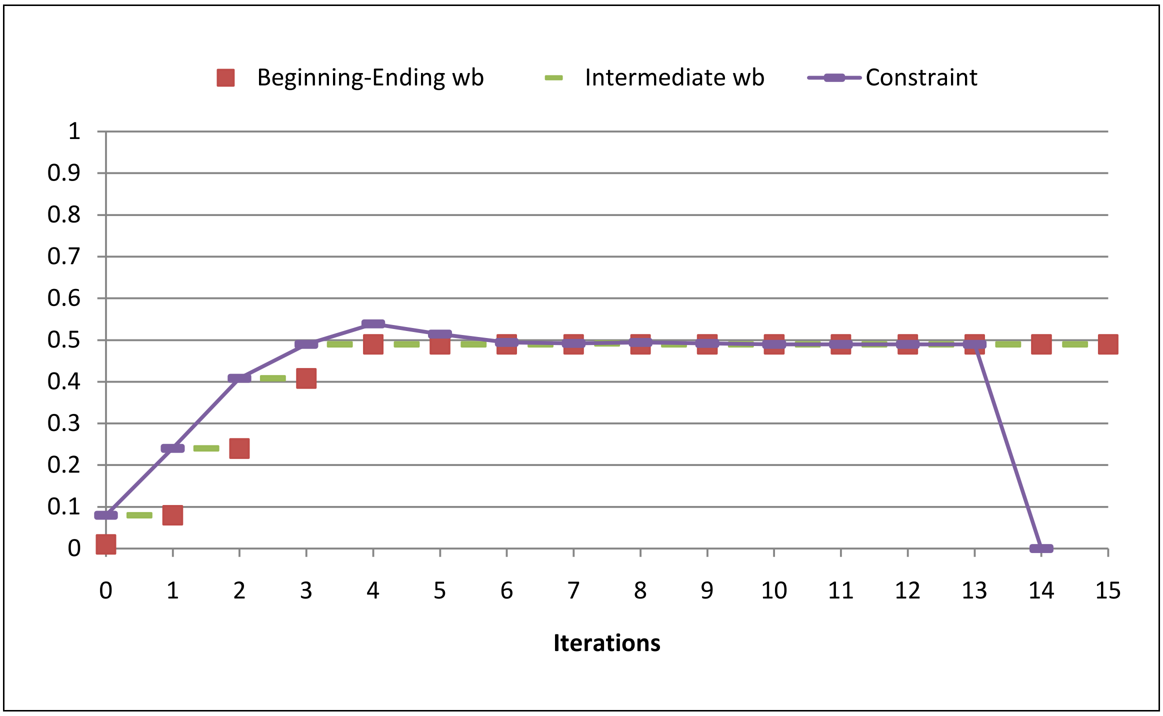


Figure 5.34: Results of Agent 2 (Group 2) in Strategy 2

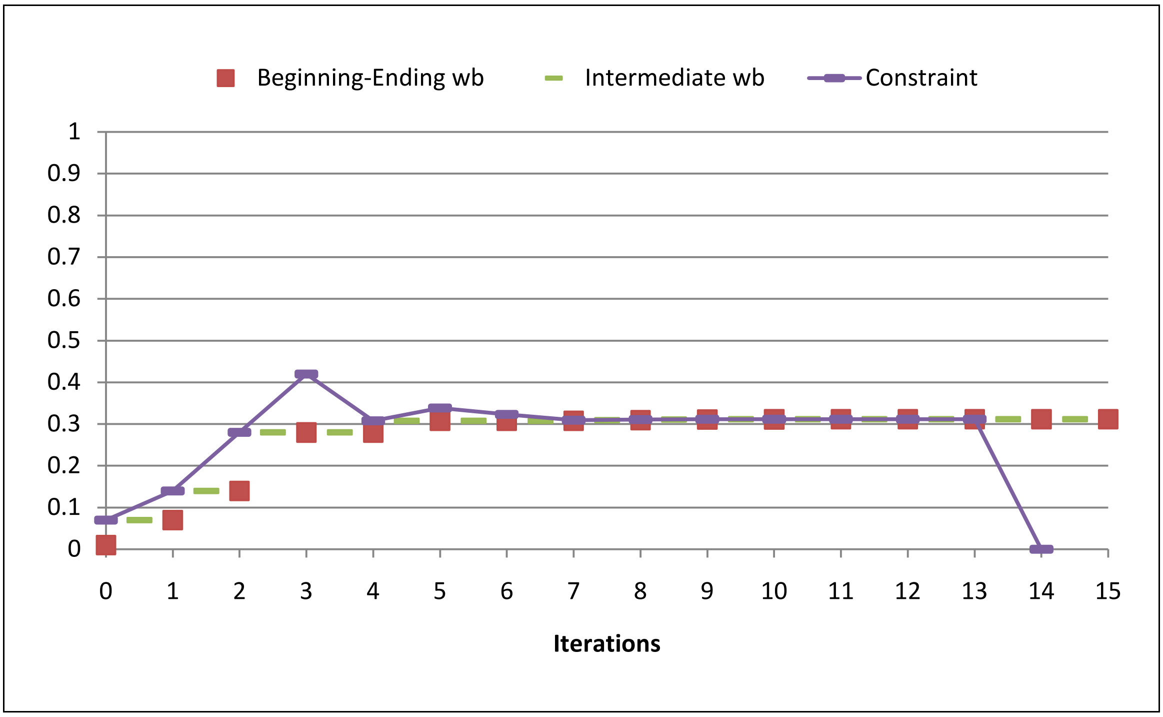


Figure 5.35: Results of Agent 3 (Group 2) in Strategy 2

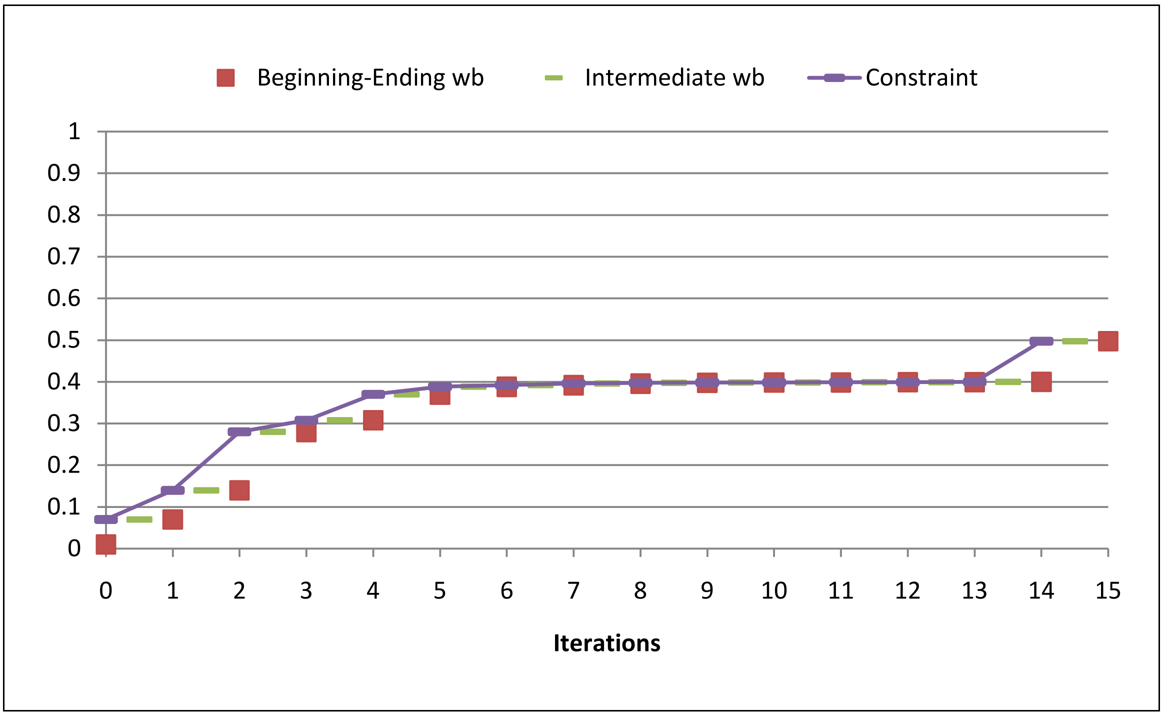


Figure 5.36: Results of Agent 4 (Group 2) in Strategy 2

5.3.2.3 Strategy 3

The results of Strategy 3 with Group 2 are shown in Figures 5.37-5.40.

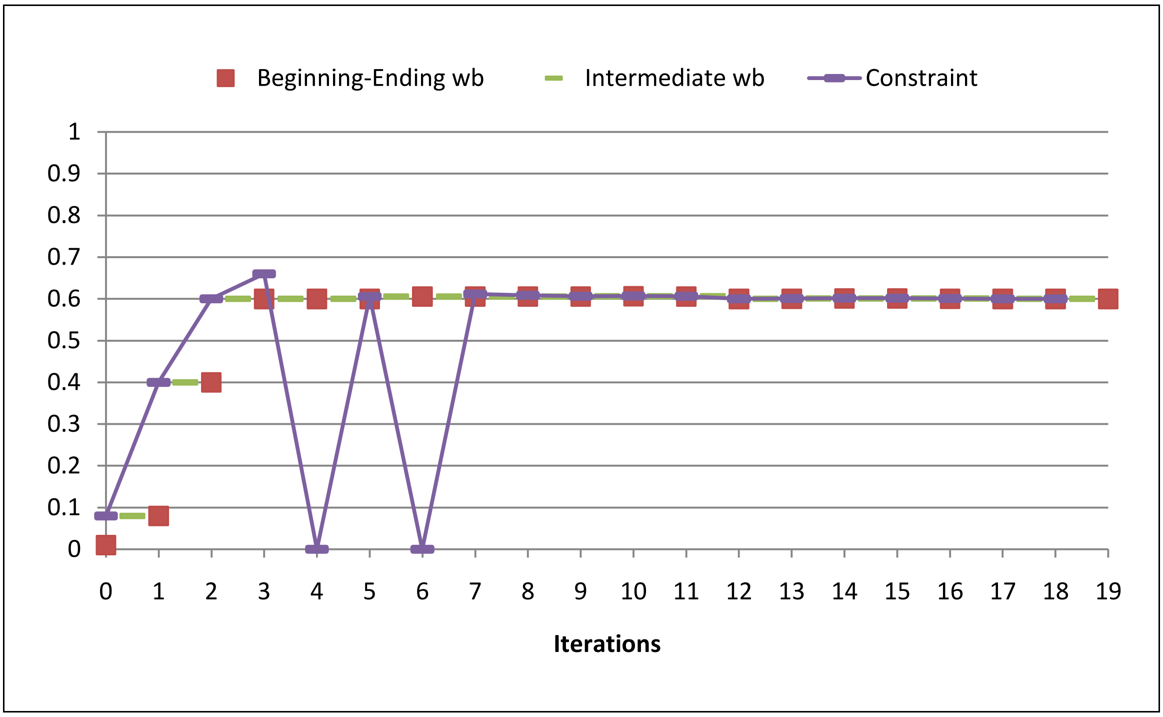


Figure 5.37: Results of Agent 1 (Group 2) in Strategy 3

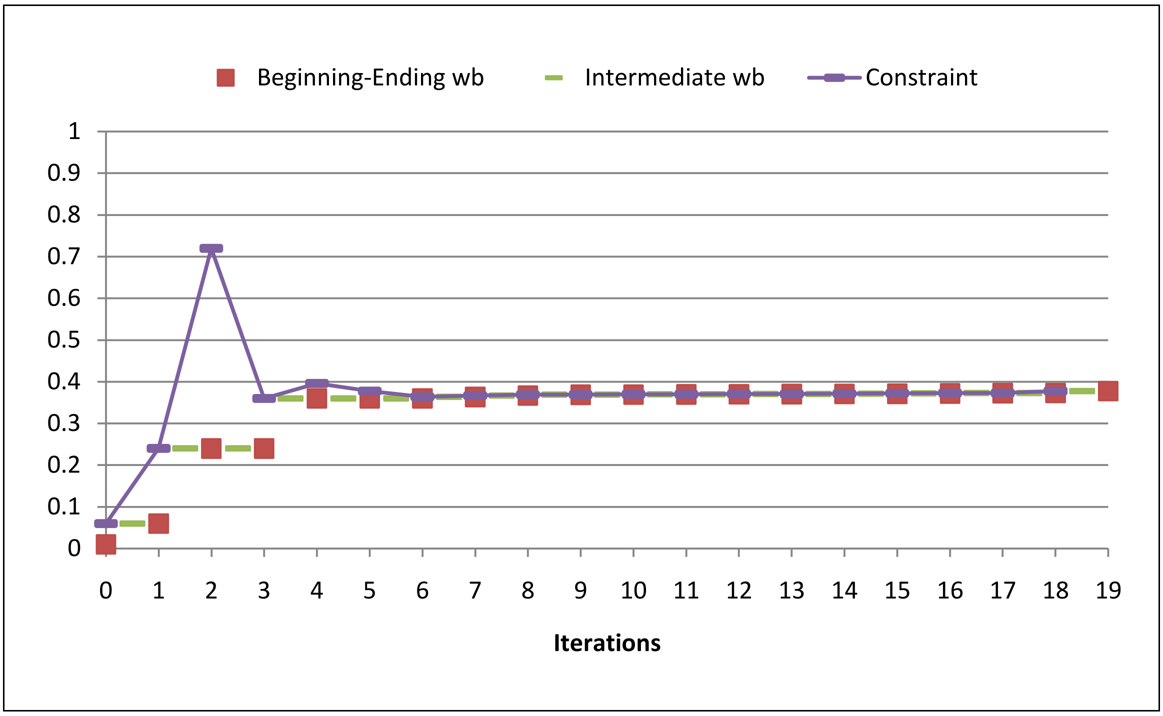


Figure 5.38: Results of Agent 2 (Group 2) in Strategy 3

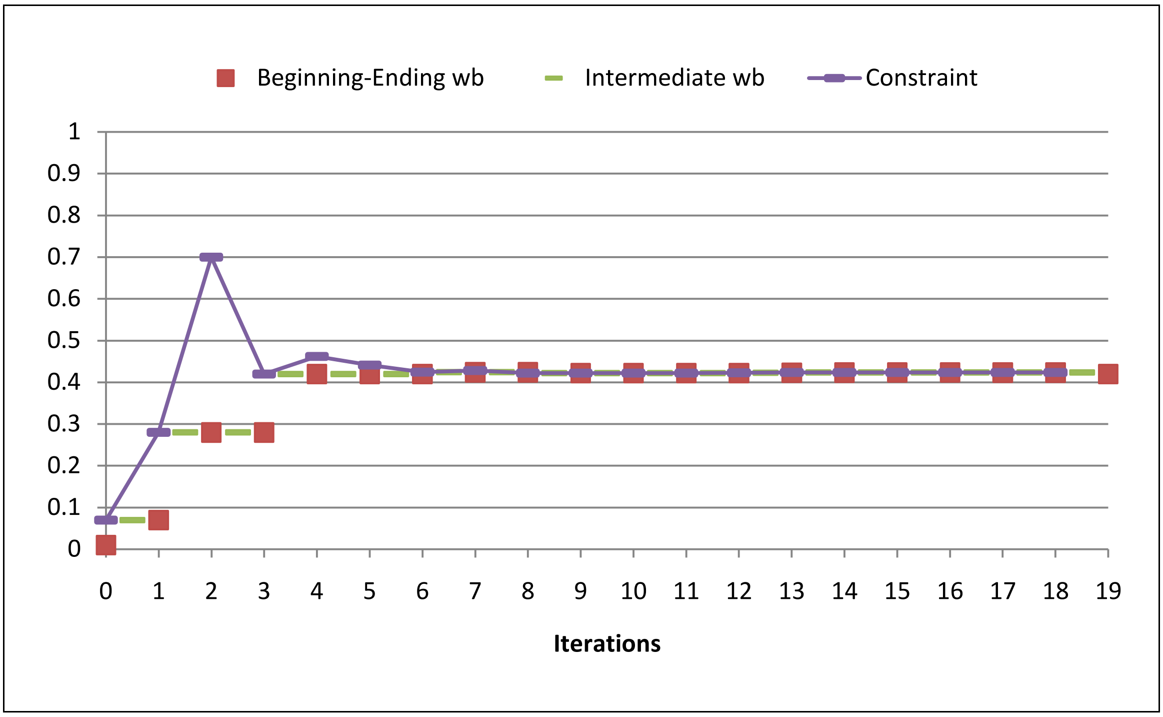


Figure 5.39: Results of Agent 3 (Group 2) in Strategy 3

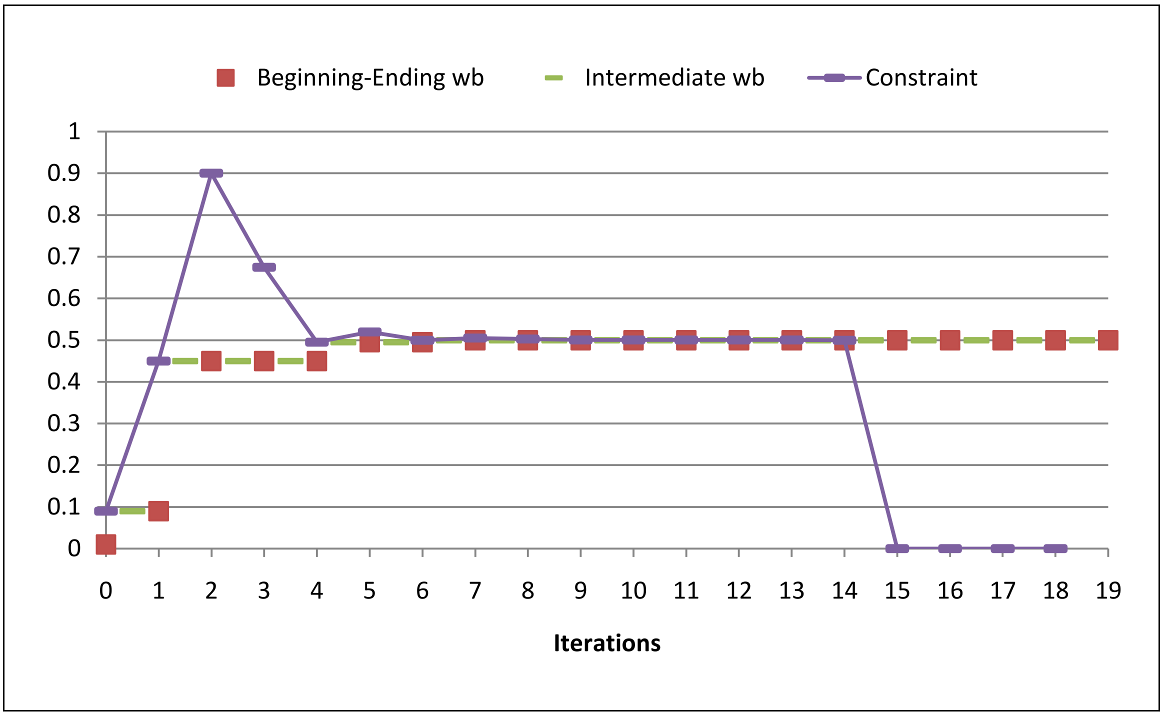


Figure 5.40: Results of Agent 4 (Group 2) in Strategy 3

5.3.2.4 Strategy 4

The results of Strategy 4 with Group 2 are shown in Figures 5.41-5.44.

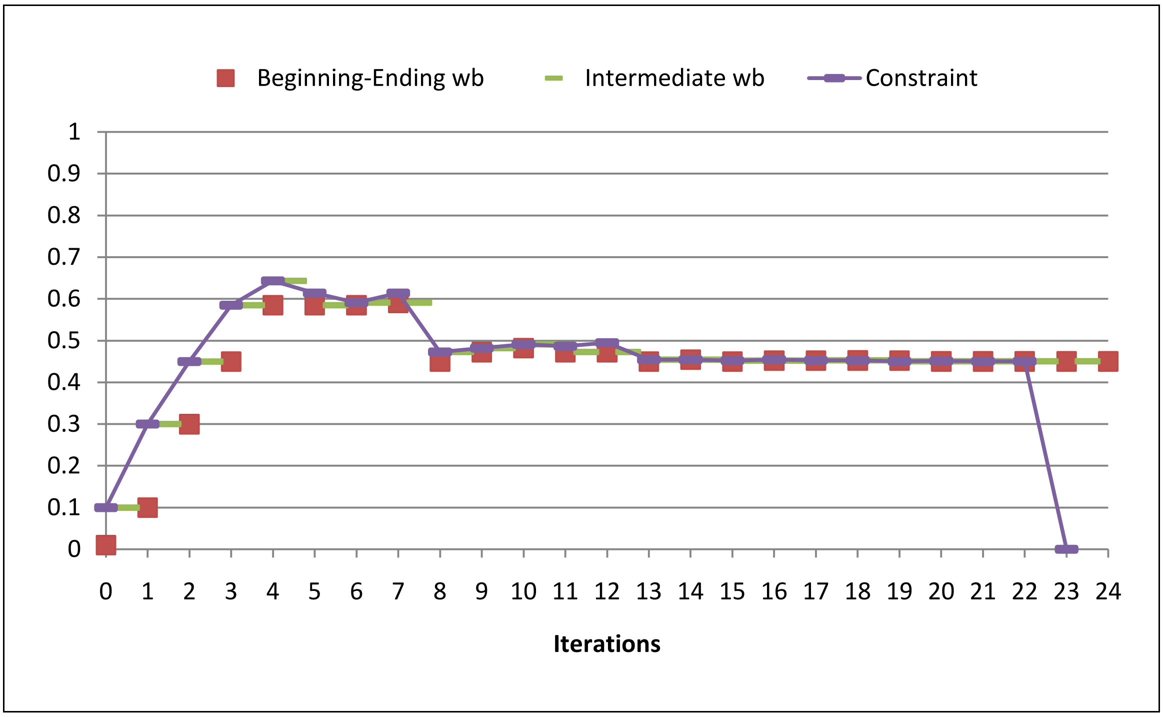


Figure 5.41: Results of Agent 1 (Group 2) in Strategy 4

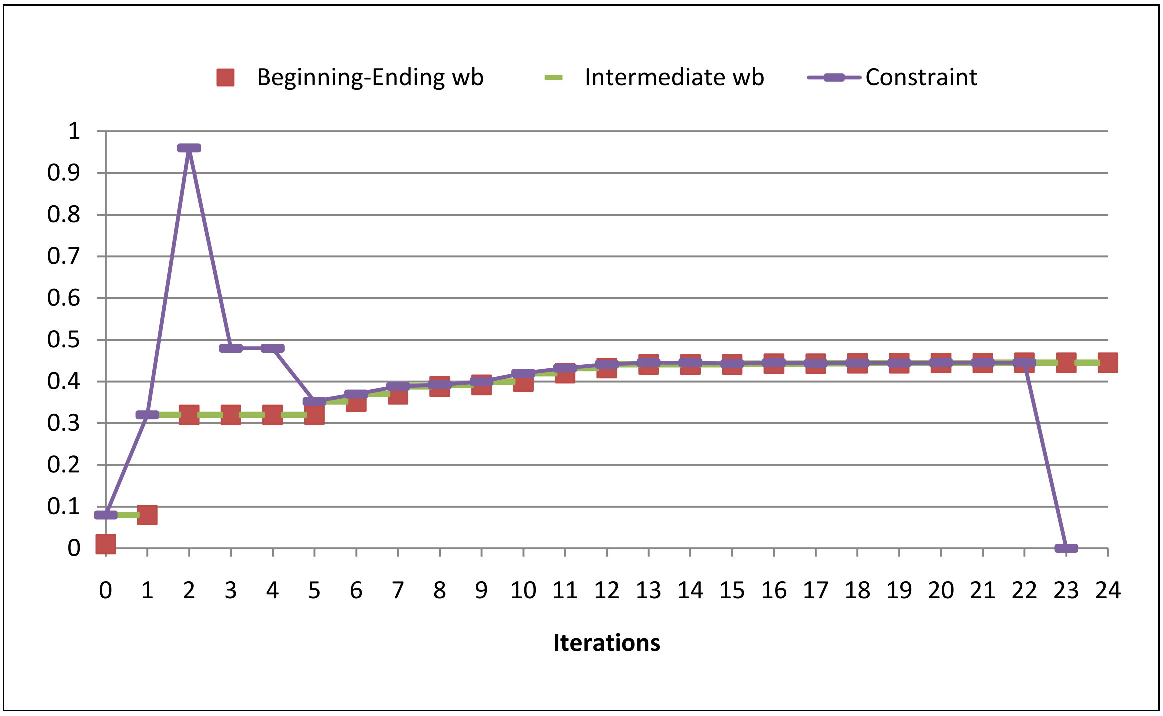


Figure 5.42: Results of Agent 2 (Group 2) in Strategy 4

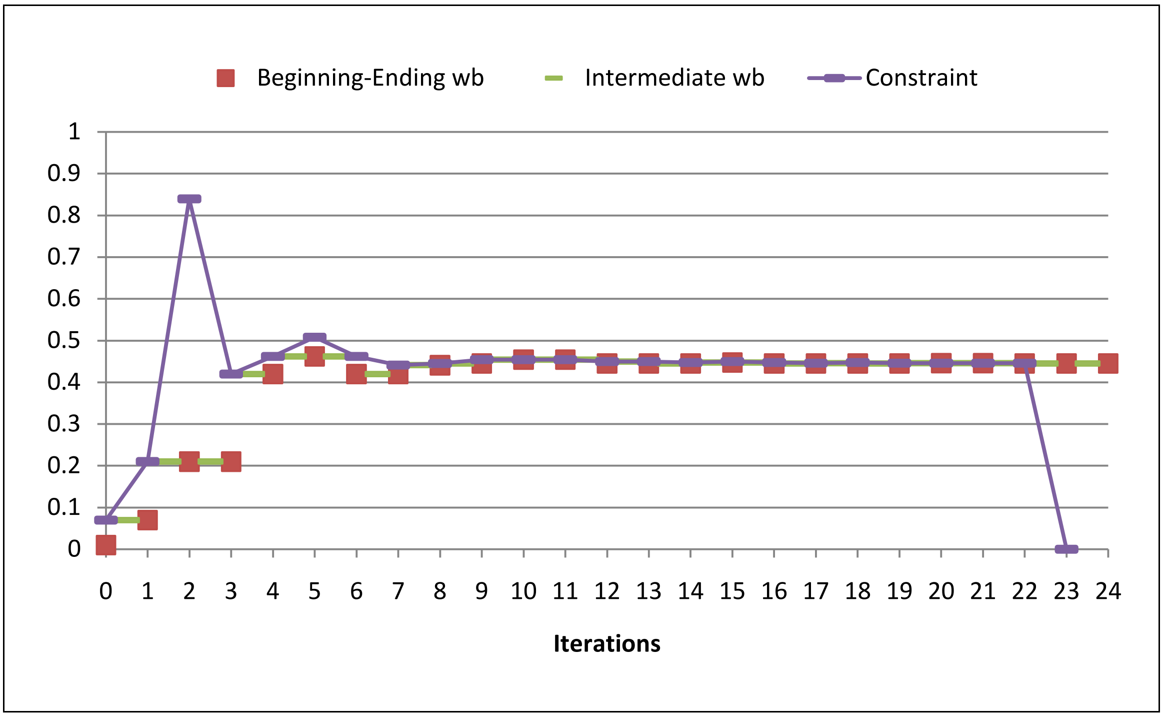


Figure 5.43: Results of Agent 3 (Group 2) in Strategy 4

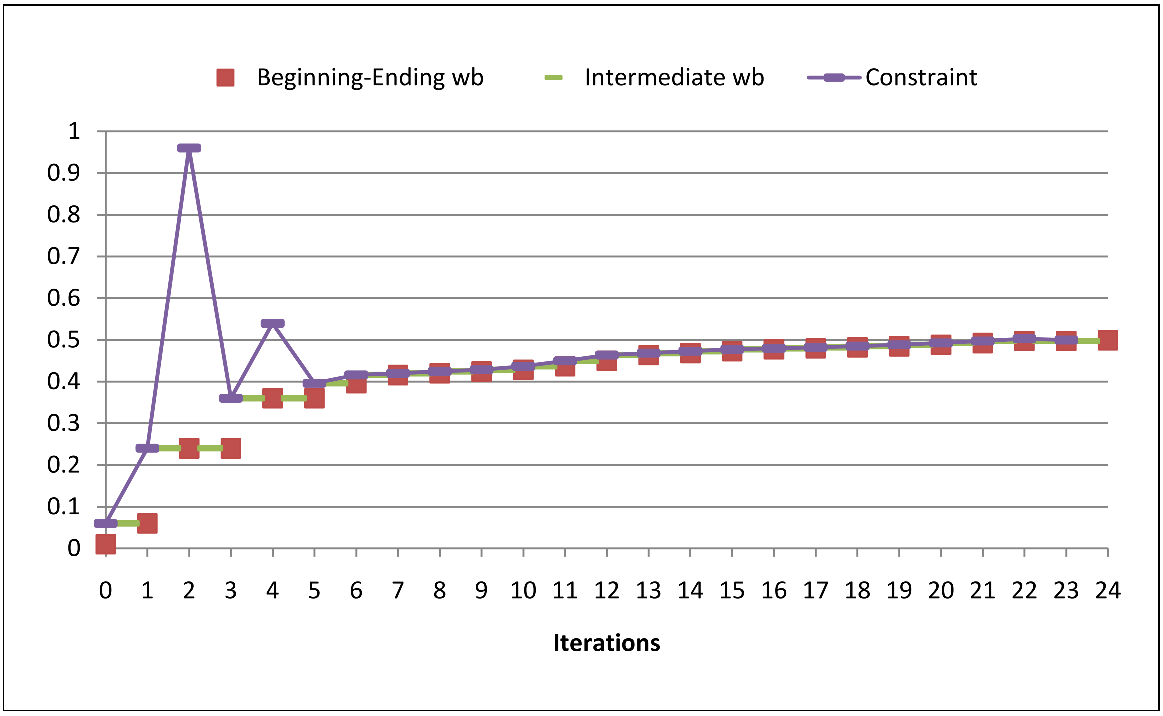


Figure 5.44: Results of Agent 4 (Group 2) in Strategy 4

5.3.2.5 Comparisons

Compared with Group 1, agents in Group 2 were more restrictive. They defined larger compromise threshold values. As seen in Figure 5.31, in Strategy 1, Agent 3 could not improve his/her wellbeing. Thus, his/her wellbeing has remained lower than the other agents during the process. In Strategies 2, 3, and 4; agents could improve their wellbeing values in a more balanced way because of the resolved conflicts. In Strategy 4, Agent 3 could even help the other agents at iterations 5 and 11 (real time observation). Total number of resolved conflicts: 5 in Strategy 4, 12 in Strategy 6, and 15 in Strategy 4.

Total satisfaction and satisfaction divergence values are compared in Figures 5.45 and 5.46. The smallest final satisfaction divergence was obtained by Strategy 4 (0.09), followed by Strategy 3 (0.339), Strategy 2 (0.415), and Strategy 1 (0.507). This order is the same as observed with Group 1. However, the order of the final total satisfaction values from largest to smallest is different than the results of Group 1. Strategies 2 and 3 resulted in the largest final total satisfaction (1.9), followed by Strategy 4 (1.84), and Strategy 1 (1.796). One may also say that they are almost similar.

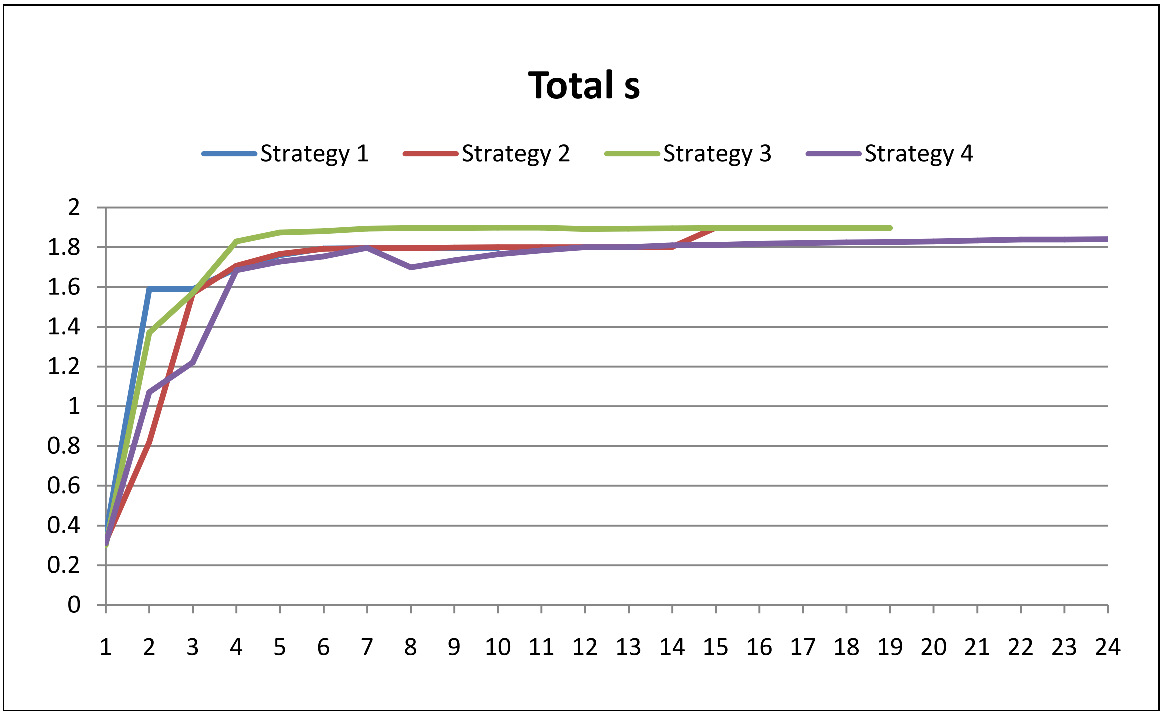


Figure 5.45: Comparison of the total satisfaction values (Group 2)

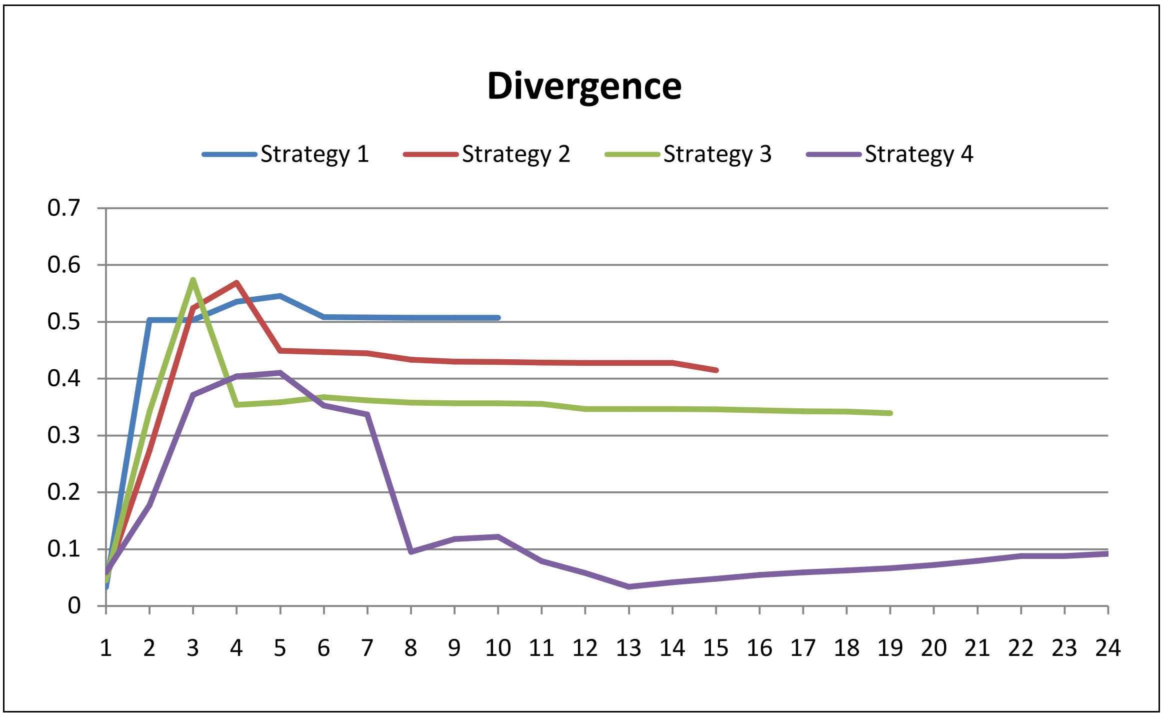


Figure 5.46: Comparison of the satisfaction divergence values (Group 2)

5.4 Conclusions

In this chapter, it is validated with human agents that promoting solidarity in the distributed design process decreases the intensity of design conflicts. While resolving design conflicts the satisfaction divergence of human agents are reduced. This result is consistent with the results of Chapter 4. As experienced with Group 1, the total satisfaction can be also increased by promoting solidarity in design. However, as experienced with Group 2, the further reduction of the satisfaction divergence can be achieved by some compromise of the total satisfaction. In Strategy 4 with Group 2, the satisfaction divergence is very small (0.09) while the total satisfaction is slightly less than Strategies 2 and 3. This is because of some concaveness on the Pareto frontier of the satisfaction space of the clutch problem. However, Strategy 4 results in larger total satisfaction on average than Strategies 3 and 2. This is shown with Monte Carlo simulation results in Figure 4.7.

As shown in Figure 5.47, if the *zero divergence line* passes through a concave part of the Pareto frontier, approximating to the *zero divergence line* increases the total satisfaction until the concaveness arises. This phenomenon is due to the nature of the design problem. Its Pareto frontier can have concaveness on the *zero divergence line* or not. If it has such concaveness, the reduction of the total satisfaction can be avoided by defining an inequality constraint on the total satisfaction. However, this will surely block to obtain a very low satisfaction divergence.

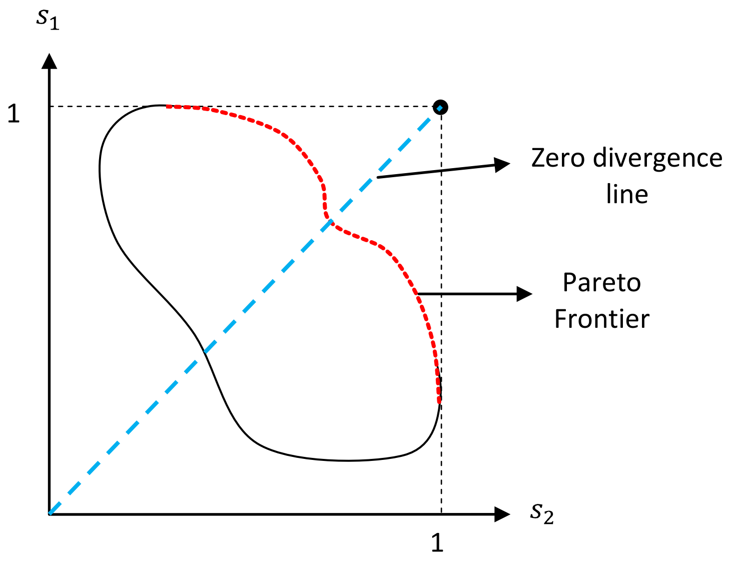


Figure 5.47: Concaveness of Pareto frontier on the zero divergence line

Some of the results of the experiments conducted with human agents (Figures 5.27, 5.28, 5.45, and 5.46) are revealed to be better than the average results of Monte Carlo simulations conducted with computer agents (Figure 4.7). Their processes ended with lower number of iterations, larger total satisfaction and smaller satisfaction divergence. This shows that human agents have defined better performing compromise threshold values and constraints than an average computer agent during the process. This may be due to human intuitions and communications. This aspect is one of those which would merit to be further investigated.

General Conclusions

The general conclusions of the dissertation are formalized as responses to the research questions.

Response to Question 1 (How to prevent design conflicts in distributed SBD while improving process performances?)

Design actors have preferences about the performance variables that they evaluate. The performance variables evaluated by their related preferences determine how the design objectives are satisfied. The satisfaction is a variable defined as an interval. The lower bound represents the minimum satisfaction while the upper bound represents the maximum satisfaction. Since design objectives of different design actors are conflicting, a decision constraint defined to increase the minimum satisfaction value of an objective will decrease the maximum satisfaction value of another objective. Thus, the convergence of satisfaction intervals is bilateral. Design actors are forced to compromise at a certain level on their satisfaction interval where maximum and minimum satisfactions are approximately equal. Design actors can define preferences on the satisfaction value in which they desire to compromise. Wellbeing indicators are developed to measure how design preferences are likely to be met at a given moment of the design process.

Wellbeing indicators provide design information at any stage of the design process. A further bottom-up design approach can be adapted where design actors make decisions directly on their wellbeing indicators. What is considered to be better or improved under epistemic uncertainty is thus precisely represented. In order to evaluate the contribution of wellbeing indicators to the design process performance, CSP simulations are performed with Monte Carlo method where design attitudes and decision sequences are random.

The results show that, with the proposed approach design actors can improve their states in wellbeing equilibrium while reducing epistemic uncertainty with consistent decision constraints on the solution space. The simulation results of a wellbeing controlled design are compared with three other process approaches that can be adapted for SBD: the process where design actors modify only their local design variables one by one, the process where design actors modify only their local design variables all-at-once, and the process where design actors modify only their performance variables. The results show that the design process performances are improved by controlling wellbeing indicators. The number of potential conflicts and the conflict intensity are reduced, because the satisfaction domination is largely avoided. This shows that design conflicts are prevented. The total satisfaction is also improved while keeping the process time minimal.

Response to Question 2 (How to model design attitudes, and prevent conflicts in distributed design systems composed of heterogeneous agents?)

A Belief-Desire-Intention (BDI) model is defined to explore the design attitudes. While upper bounds of the design performance intervals represent the possible best cases, their lower bounds represent possible worst cases. These cases reflect the beliefs of the design agent about how its design performance variables converge towards design objectives. The convergence of the performance variable of an agent is bilateral in a conflicting design problem. This convergence depends on the unpredictable reactions of the other design agents, since design agents are considered as heterogeneous. The design agent therefore identifies some preferences about its performance variables and satisfaction intervals considering its initial beliefs. These preferences reflect the agent's desires towards the uncertain convergence. During the design process, the design agent can state intentions for increasing its satisfaction from the dynamic design model by evaluating how its instant beliefs fulfill its desires. The design agent reacts to uncertainties by defining decision constraints that restrict the solution space, with the aim of improving its worst cases. The intentions are reflected with how frequently and how restrictively the decision constraints are defined during the design process.

The character of a design agent can be evaluated by the BDI model. The interactions of design agents with heterogeneous characters can thus be simulated. As an extended bottom-up design approach, the design attitudes of heterogeneous designers in distributed design are evaluated beforehand. In this approach, design agents can make trade-off intentions on their wellbeing values derived from their beliefs and desires.

CSP simulations are performed with Monte Carlo method where designer characters are defined randomly. The simulation results show again that the performance of the design process is significantly increased with this approach. In this approach, designer dominations caused by the process itself are eliminated. The increase of the potential conflicts and the conflict intensity is thus prevented. The simulation results show also that moderate reciprocal altruism decreases the conflict intensity, while too much altruism can decrease the total satisfaction obtained from the final solution.

Response to Question 3 (How to justify and resolve design conflicts in distributed SBD?)

A conflict management model is developed for design conflicts that cannot be avoided. It consists of conflict justification and conflict resolution. If a constraint is rejected it represents a potential conflict, since the desires of the agent that defines the constraint are not satisfied. If there is another agent in a better wellbeing state, the conflict is justified. This is because it is considered that the shared solution space is not restricted in equilibrium. At least one agent has restricted the space more than the agent in conflict. A CoCSP model is developed in order to resolve the justified conflicts. In this model, the other agents can help the agent in conflict to enable the rejected constraint. This is performed by relaxing some of their constraints from their list of accepted constraints. The model detects which agent can help and how it can help optimally. It is composed of three phases:

- Negotiation phase: all the help possibilities are detected.
- Testing phase: feasibilities of different help possibilities are tested with CP techniques and the optimal help among the feasible help solutions is detected.
- Approval phase: the approval of the optimal help is requested.

Monte Carlo simulations of this model are performed with heterogeneous agent characters defined randomly. Also some experiments are conducted with human agents. The results validate that the intensity of the conflicts are lowered when the conflicts are resolved by this model.

Response to Question 4 (How promoting solidarity is useful in distributed design?)

The conflict resolution system can be adopted for different strategies which take into account the solidarity architecture of design agents. The Monte Carlo simulations of computer agents and the experiments conducted with human agents show that promoting solidarity reduces the conflict intensity. However, this gain is obtained at the cost of the increase of design process time, because conflict resolution causes loopbacks. Additionally, promoting solidarity can increase the total satisfaction if the Pareto frontier of the satisfaction space remains convex while approximating to the zero divergence line.

Limitations and Future works

The limitations of this work are as follows:

- It is limited to measurable design problems where all the design aspects can be represented as numerical variables.
- Variables are evaluated by the length of their intervals. However, the density of the consistent values that can be assigned to a variable may not be uniform along the interval. While reducing the length of an interval, we may not eliminate any solution.
- There are no guidelines proposed to designers about how to define optimal threshold of compromise and coefficient of restriction values. This is because each design problem is unique, and the optimal values will be different.
- Feasibility of the conflict resolution is not guaranteed. During a design process, it may happen that design agents want to help, but the model cannot detect a feasible help. This depends on the constraints that have been previously defined by the agent that would help. An agent can reduce the same length of its interval by only one constraint or several constraints. If the number of the constraints reducing the length of the interval is not sufficient enough, there may not be any constraint combination that can be removed to provide help.

In future works, the feasibility of the conflict resolution can be increased by suggesting agents (that would like to help) to increase their number of constraints on the eliminated part of their intervals. More constraint combinations would thus be provided so that the possibility to detect a feasible help could be increased. Some guidelines can also be provided to detect better desires and intentions. Further experiments can be conducted to test the models extensively through human interactions. Another future work should evaluate the density of intervals.

Final words and personal assessment of model utility

The final model emerging from this work is novel and presents a complete framework to improve the distributed design process with practical applications. Integrating the model into a computer-aided design (CAD) software package would enhance the CAD's utility. Contrarily to the conventional CAD where the product dimensioning is not concurrent, our model provides a mechanism for concurrently dimensioning a product by uncertainty reduction. This would also avoid iterative loopbacks since the product model is not represented with a deterministic concept, but with a palette of many concepts: SBD principle. Designers might also evaluate their absolute and relative states through wellbeing indicators during the design process. By providing these personal indicators to other designers, this would allow the designers to better understand and help each other (negotiable CAD). Further works could consist in deploying and testing a distributed CAD with our model through group sociology approaches. In this type of distributed CAD, design conflicts could thus be explicitly revealed and resolved.

Personal Papers

These are my papers published or submitted during my PhD studies.

Journal Papers:

- Canbaz B., Yannou B., Yvars P.-A. (2013) “Improving design process performance of distributed design systems with controlling wellbeing indicators of design actors”, **submitted to** Journal of Mechanical Design.

- Canbaz B., Yannou B., Yvars P.-A. (2013) “Preventing design conflicts in distributed design systems composed of heterogeneous design agents”, **submitted to** Engineering Applications of Artificial Intelligence.

- Canbaz B., Yannou B., Yvars P.-A. (2013) “Resolving design conflicts and promoting solidarity in distributed design”, **submitted to** IEEE Transactions on Systems, Man, and Cybernetics: Systems.

International Conference Papers:

- Canbaz B., Yannou B., Yvars P.-A., (2011) “A new framework for collaborative set-based design: Application to the design problem of a hollow cylindrical cantilever beam”, in *Proceedings of ASME IDETC 2011: International Design Engineering Technical Conferences*, Washington, D.C., USA, August 28-31

- Canbaz B., Yannou B., Yvars P.-A. (2012) “Constraint programming simulation of a distributed set-based design framework with control indicator”, in *Proceedings of ASME IDETC 2012: International Design Engineering Technical Conferences*, Chicago, IL, USA, August 12-15

- Canbaz B., Yannou B., Yvars P.-A. (2013) “Expanding the bottom-up design approach through integrating design attitudes into set-based design”, in *Proceedings of ASME IDETC 2013: International Design Engineering Technical Conferences*, Portland, OR, USA, August 4-7

References

Allison, J.T., Kokkolaras, M., Papalambros, P.Y., 2006. On Selecting Single-Level Formulations for Complex System Design Optimization. J. Mech. Des. 129, 898–906.

Antonsson, E.K., Otto, K.N., 1995. Imprecision in Engineering Design. Journal of Mechanical Design 117, 25–32.

Braha, D., Maimon, O., 1998. The measurement of a design structural and functional complexity. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 28, 527–535.

Bratman, M.E., Israel, D.J., Pollack, M.E., 1988. Plans and resource-bounded practical reasoning. Computational Intelligence 4, 349–355.

Brazier, F.M.T., van Langen, P.H.G., Treur, J., 1995. Modelling conflict management in design: An explicit approach. AI EDAM 9, 353–366.

Ceroni, J.A., Velásquez, A.A., 2003. Conflict detection and resolution in distributed design. Production Planning & Control 14, 734–742.

Chanron, V., Lewis, K., 2005. A study of convergence in decentralized design processes. Res Eng Design 16, 133–145.

Cooper, A.B., Georgiopoulos, P., Kim, H.M., Papalambros, P.Y., 2005. Analytical Target Setting: An Enterprise Context in Optimal Product Design. J. Mech. Des. 128, 4–13.

Costantino, C.A., Merchant, C.S., 1996. Designing conflict management systems: a guide to creating productive and healthy organizations. Jossey-Bass Publishers.

Détienne, F., Boujut, J.-F., Hohmann, B., 2004. Characterization of Collaborative Design and Interaction Management Activities in a Distant Engineering Design Situation, in: Darses, F., Dieng, R., Simone, C., Zacklad, M. (Eds.), Cooperative Systems Design. IOS Press, Amsterdam, Netherlands, pp. 83–98.

Devendorf, E., Lewis, K., 2011. The Impact of Process Architecture on Equilibrium Stability in Distributed Design. Journal of Mechanical Design 133, 101001.

Djaouti, D., Alvarez, J., Jessel, J.-P., Rampnoux, O., 2011. Origins of Serious Games, in: Ma, M., Oikonomou, A., Jain, L.C. (Eds.), Serious Games and Edutainment Applications. Springer London, pp. 25–43.

Fathianathan, M., Panchal, J.H., 2009. Modelling an ongoing design process utilizing top-down and bottom-up design strategies. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 223, 547–560.

Favela, J., Wong, A., Chakravarthy, A., 1993. Supporting collaborative engineering design. Engineering with Computers 9, 125–132.

Ganguly, S., Wu, T., Blackhurst, J., 2008. A Price-Based Negotiation Mechanism for Distributed Collaborative Design. IEEE Transactions on Engineering Management 55, 496–507.

Granvilliers, L., 2012. Adaptive Bisection of Numerical CSPs, in: Milano, M. (Ed.), Principles and Practice of Constraint Programming, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 290–298.

Gray, B., 1990. Collaborating: Finding Common Ground for Multiparty Problems. The Academy of Management Review 15, 545.

Harris, D., H., 1994. Productivity Linkages in Computer-Aided Design, in: Organizational Linkages: Understanding the Productivity Paradox. pp. 240–261.

IBM, 2012. IBM ILOG CPLEX CP Optimizer for Constraint Programs - Features and benefits [WWW Document]. URL http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/about/

Jabrouni, H., Kamsu-Foguem, B., Geneste, L., Vaysse, C., 2011. Continuous improvement through knowledge-guided analysis in experience feedback. Engineering Applications of Artificial Intelligence 24, 1419–1431.

Jagannatha Rao, J.R., Badhrinath, K., Pakala, R., Mistree, F., 1997. A Study of Optimal Design Under Conflict Using Models of Multi-Player Games. Engineering Optimization 28, 63–94.

Kelly, J.C., Maheut, P., Petiot, J.-F., Papalambros, P.Y., 2011. Incorporating user shape preference in engineering design optimisation. Journal of Engineering Design 22, 627–650.

Kim, H.M., Michelena, N.F., Papalambros, P.Y., Jiang, T., 2003. Target Cascading in Optimal System Design. Journal of Mechanical Design 125, 474.

Klein, M., 1991. Supporting conflict resolution in cooperative design systems. IEEE Transactions on Systems, Man and Cybernetics 21, 1379–1390.

Klein, M., Sayama, H., Faratin, P., Bar-Yam, Y., 2003. The Dynamics of Collaborative Design: Insights from Complex Systems and Negotiation Research. Concurrent Engineering 11, 201–209.

Koulinitch, A.S., Sheremetov, L.B., 1998. Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor. Expert Systems with Applications 15, 295–307.

Krishnamachari, R.S., Papalambros, P.Y., 1997. Hierarchical Decomposition Synthesis in Optimal Systems Design. J. Mech. Des. 119, 448–457.

Kwon, O.B., Lee, K.C., 2002. MACE: multi-agents coordination engine to resolve conflicts among functional units in an enterprise. Expert Systems with Applications 23, 9–21.

Lee, H., Whang, S., 1999. Decentralized Multi-Echelon Supply Chains: Incentives and Information. Management Science 45, 633–640.

Lewis, K., Mistree, F., 1998. Collaborative, Sequential, and Isolated Decisions in Design. Journal of Mechanical Design 120, 643.

Lewis, K., Mistree, F., 2001. Modeling Subsystem Interactions: A Game Theoretic Approach. Journal of Design and Manufacturing Automation 1, 17–35.

Liu, H., Chen, W., Scott, M.J., Qureshi, K., 2008. Determination of ranged sets of design specifications by incorporating design-space heterogeneity. Engineering Optimization 40, 1011–1029.

Lu, S.C.-Y., Cai, J., Burkett, W., Udwadia, F., 2000. A Methodology for Collaborative Design Process and Conflict Analysis. CIRP Annals - Manufacturing Technology 49, 69–73.

Malak, R.J., Aughenbaugh, J.M., Paredis, C.J.J., 2009. Multi-attribute utility analysis in set-based conceptual design. Computer-Aided Design 41, 214–227.

Marfisi-Schottman, I., George, G., Tarpin-Bernard, F., 2010. Tools and Methods for Efficiently Designing Serious Games, in: 4th Europeen Conference on Games Based Learning ECGBL2010. Copenhagen, Denmark,, pp. 226–234.

Martin, J.D., Simpson, T.W., 2006. A Methodology to Manage System-level Uncertainty During Conceptual Design. J. Mech. Des. 128, 959–968.

Messac, A., 1996. Physical programming - Effective optimization for computational design. AIAA Journal 34, 149–158.

Meyer, Y., Yvars, P.-A., 2012. Optimization of a passive structure for active vibration isolation: an interval-computation- and constraint-propagation-based approach. Engineering Optimization 44, 1463–1489.

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

Nahm, Y.-E., Ishikawa, H., 2004. Integrated Product and Process Modeling for Collaborative Design Environment. Concurrent Engineering 12, 5–23.

Novak, S., Eppinger, S.D., 2001. Sourcing By Design: Product Complexity and the Supply Chain. Management Science 47, 189–204.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H., 2007. Product Development Process, in: Wallace, K., Blessing, L. (Eds.), Engineering Design: A Systematic Approach. Springer London, pp. 125–143.

Panchal, J.H., Fernández, M.G., Paredis, C.J.J., Allen, J.K., Mistree, F., 2007. An Interval-based Constraint Satisfaction (IBCS) Method for Decentralized, Collaborative Multifunctional Design. Concurrent Engineering 15, 309–323.

Papalambros, P.Y., Michelena, N.F., Kikuchi, N., 1997. Distributed Cooperative Systems Design, in: Proceedings of the 11th International Conference on Engineering Design. Tampere, Finland, pp. 265–270.

Parry, G.W., 1996. The characterization of uncertainty in Probabilistic Risk Assessments of complex systems. Reliability Engineering & System Safety 54, 119–126.

Parsons, M.G., Singer, D.J., Sauter, J.A., 1999. A hybrid agent approach for set-based conceptual ship design, in: Proceedings of 10th International Conference on Computer Applications in Shipbuilding. Cambridge, MA.

Pelled, L.H., Eisenhardt, K.M., Xin, K.R., 1999. Exploring the Black Box: An Analysis of Work Group Diversity, Conflict, and Performance. Administrative Science Quarterly 44, 1.

Petiot, J.-F., Dagher, A., 2011. Preference-oriented form design: application to cars' headlights. Int J Interact Des Manuf 5, 17–27.

Rao, S.S., Freiheit, T.I., 1991. A Modified Game Theory Approach to Multiobjective Optimization. J. Mech. Des. 113, 286–291.

Schlosser, J., Paredis, C.J.J., 2007. Managing Multiple Sources of Epistemic Uncertainty in Engineering Decision Making (SAE Technical Paper No. 2007-01-1481). SAE International, Warrendale, PA.

Simpson, T.W., Martins, J.R.R.A., 2011. Multidisciplinary Design Optimization for Complex Engineered Systems: Report From a National Science Foundation Workshop. J. Mech. Des. 133, 101002–101002.

Simpson, T.W., Rosen, D., Allen, J.K., Mistree, F., 1998. Metrics for Assessing Design Freedom and Information Certainty in the Early Stages of Design. J. Mech. Des. 120, 628–635.

Sobek, D.K., Ward, A.C., Liker, J., 1999. Toyota's Principles of Set-Based Concurrent Engineering. Sloan Management Review 40, 67–83.

Sobieszczanski-Sobieski, J., Barthelemy, J.F.M., Giles, G.L., 1984. Aerospace engineering design by systematic decomposition and multilevel optimization, in: 14th Congr. of the International Council of the Aeronautical Sciences (ICAS). Toulouse, France.

Sobieszczanski-Sobieski, J., Haftka, R.T., 1997. Multidisciplinary aerospace design optimization: survey of recent developments. Structural Optimization 14, 1–23.

Suh, N.P., 1999. A Theory of Complexity, Periodicity and the Design Axioms. Research in Engineering Design 11, 116–132.

Vareilles, É., Aldanondo, M., Codet de Boisse, A., Coudert, T., Gaborit, P., Geneste, L., 2012. How to take into account general and contextual knowledge for interactive aiding design: Towards the coupling of CSP and CBR approaches. Engineering Applications of Artificial Intelligence 25, 31–47.

Vincent, T.L., 1983. Game Theory as a Design Tool. J. Mech. Des. 105, 165–170.

Wall, J.A., Callister, R.R., 1995. Conflict and Its Management. Journal of Management 21, 515–558.

Wang, G.G., 2003. Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points. J. Mech. Des. 125, 210–220.

Wang, J., Terpenny, J., 2003. Interactive evolutionary solution synthesis in fuzzy set-based preliminary engineering design. Journal of Intelligent Manufacturing 14, 153–167.

Ward, A.C., Liker, J., Sobek, D.K., Cristiano, J.J., 1994. Set-based concurrent engineering and Toyota, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 79–90.

Wujek, B.A., Renaud, J.E., Batill, S.M., Brockman, J.B., 1996. Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing Environment. Concurrent Engineering 4, 361–377.

Xiao, A., Zeng, S., Allen, J.K., Rosen, D.W., Mistree, F., 2005. Collaborative multidisciplinary decision making using game theory and design capability indices. Res Eng Design 16, 57–72.

Yang, M.C., Iii, W.H.W., Cutkosky, M.R., 2005. Design information retrieval: a thesauri-based approach for reuse of informal design information. Engineering with Computers 21, 177–192.

Yannou, B., 2004. Managing uncertainty of product data. An enhancement on constraint programming techniques, in: Tichkiewitch, S., Brissaud, D. (Eds.), Methods and Tools for Co-operative and Integrated Design Methods and Tools for Co-operative and Integrated Design. Kluwer Academic Publishers, Springer, pp. 195–208.

Yannou, B., Harmel, G., 2004. A Comparative Study of Constraint Programming Techniques Over Intervals in Preliminary Design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 189–198.

Yannou, B., Harmel, G., 2006. Use of Constraint Programming for Design, in: ElMaraghy, H.A., ElMaraghy, W.H. (Eds.), Advances in Design, Springer Series in Advanced Manufacturing. Springer London, pp. 145–157.

Yannou, B., Mazur, C., Yvars, P.-A., 2010. Parameterization and dimensioning of the multi-disc clutch in the CO4 environment, Technical report (Cahier d'Etudes et de Recherches), Ecole Centrale Paris, September 7th 2010, available at “http://www.lgi.ecp.fr/uploads/Recherche/CO4Multi_DiscClutch_Technical_Document” on July 9th 2013.

Yannou, B., Yvars, P.-A., Hoyle, C., Chen, W., 2013. Set-based design by simulation of usage scenario coverage. Journal of Engineering Design. DOI:10.1080/09544828.2013.780201.

Yvars, P.-A., 2009. A CSP approach for the network of product lifecycle constraints consistency in a collaborative design context. Engineering Applications of Artificial Intelligence 22, 961–970.

Yvars, P.-A., 2010. A Constraint Based Decision Support System for Deadlocks Resolution in Collaborative New Product Design. Journal of Decision Systems 19, 57–74.

Zhang, X., Huang, H.-Z., Wang, Z., Liu, Y., Xu, H.-W., 2011. A New Method for Achieving Flexibility in Hierarchical Multilevel System Design. Concurrent Engineering 19, 187–196.

Zhao, L., Jin, Y., 2003. Work Structure Based Collaborative Engineering Design, in: Proceedings of ASME Design Engineering Technical Conferences. ASME, pp. 865–874.

Zheng, Y., Shen, H., Sun, C., 2011. Collaborative design: Improving efficiency by concurrent execution of Boolean tasks. Expert Systems with Applications 38, 1089–1098.

Appendices

Appendix A: Multi-disc Clutch System

The system consists of a weight to be lifted by a hoisting drum, an engine, a gear box, and a clutch between the hoisting drum and the gear box. This system is shown in Figure A.1. P1 is Position 1, P2 is Position 2, and P3 is position 3 for stress and safety calculations.

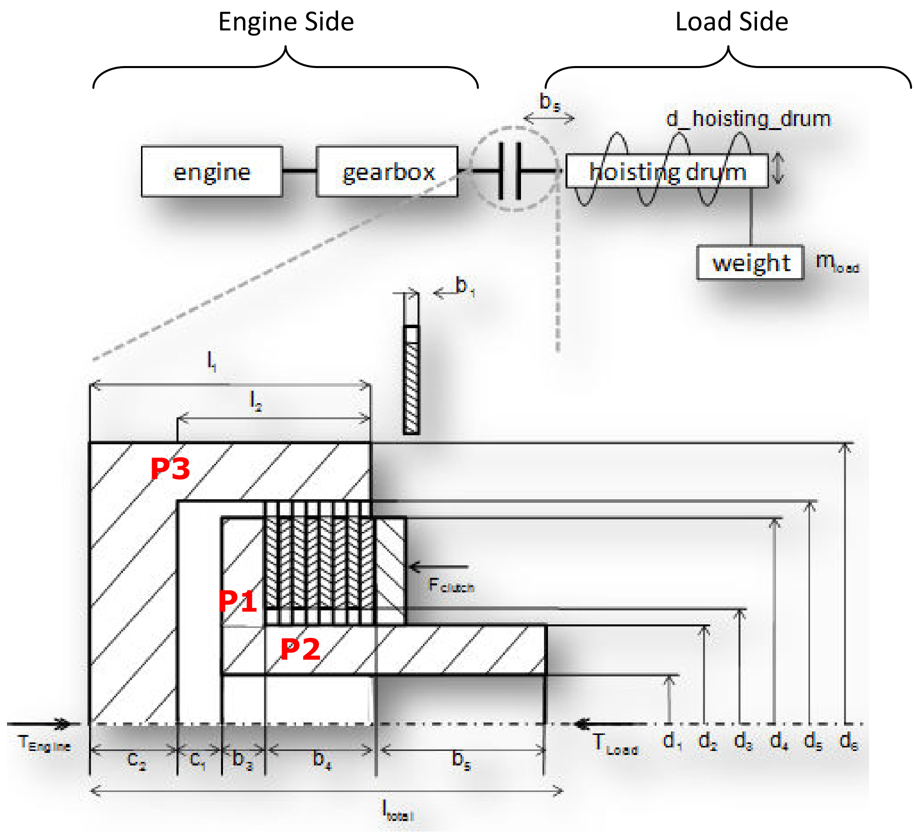


Figure A.1: Multi-disc clutch system

The constant values are listed in Table A.1.

Table A.1: Initial Design choices already made by the designers

Space between chassis and shaft	c_1 (mm) = 10
Thickness of chassis plate	c_2 (mm) = 30
Length of shaft between clutch and hoisting drum	b_5 (mm) = 300
Gravity	$g\left(\frac{N}{kg}\right) = 9{,}81$
Temperature of environment	$temp_0$ (°C) = 20
Diameter of hoisting drum	$d_{hoisting\_drum}\ (mm) = 500$

Three different materials can be chosen for the chassis and shaft of the clutch: aluminum, cast iron and steel. The specifications of these materials are listed in Table A.2.

Table A.2: Choice of material for chassis and shaft

	Aluminum	Cast iron	Steel
Stiffness (N/mm²)	$\sigma_{\max} = 150$	$\sigma_{\max} = 300$	$\sigma_{\max} = 500$
Density (kg/m³)	$\rho_1 = 2700$	$\rho_1 = 7100$	$\rho_1 = 7850$
Thermal conductivity (W/mK)	$\lambda = 237$	$\lambda = 45$	$\lambda = 30$

Two different friction materials can be chosen for clutch discs / lamellae (Table 3). The specifications of these materials are listed in Table A.3.

Table A.3: Choice of material for clutch discs / lamellae

	Material 1	**Material 2**
$Friction\ value\ \ (no\ unit)$	$\mu = 0.38$	$\mu = 0.45$
$Density\ \left(\frac{kg}{m^3}\right)$	$\rho_1 = 1740$	$\rho_1 = 2400$
$Max\ friction\ speed\ \left(\frac{m}{s}\right)$	$v_{relative\_max} = 30$	$v_{relative\_max} = 15$
$Max\ allowed\ temperature\ (°C)$	$temp_{max} = 200$	$temp_{max} = 250$
$Max\ allowed\ disc\ pressure\ \left(\frac{N}{mm^2}\right)$	$pressure_{max} = 1$	$pressure_{max} = 2$
$Max\ relative\ friction\ work\ \left(\frac{W}{mm^2}\right)$	$q_{max} = 2$	$q_{max} = 4$

Three different engines can be chosen. The type of engine defines the maximal weight to be lifted. The specifications of these engines are listed in Table A.4.

Table A.4: Choice of engine

	Engine 1	Engine 2	Engine 3
$Torque\ (Nm)$	$T_{engine} = 150$	$T_{engine} = 350$	$T_{engine} = 200$
$Number\ of\ revolutions\ \ (min^{-1})$	$n_{engine\_0} = 1500$	$n_{engine\_0} = 800$	$n_{engine\_0} = 1100$
$Weight\ \ (kg)$	$m_{engine} = 110$	$m_{engine} = 270$	$m_{engine} = 220$
$Moment\ of\ inertia\ (kgm^2)$	$J_{engine} = 0.13$	$J_{engine} = 0.35$	$J_{engine} = 0.20$

A friction multi-disc clutch consists of at least an outer and an inner lamella. The outer lamella is on the chassis side, and the inner lamella is on the shaft side. Figure A.2 shows a pair of lamellae. The outer lamella is defined by d_5 and d_3, while the inner lamella is defined by d_4 and d_2. d_m is the intermediate diameter of the friction surface. Three different discs, each one consisting of different sizes of lamella, can be chosen. Specifications of different classes of discs are shown in Table A.5.

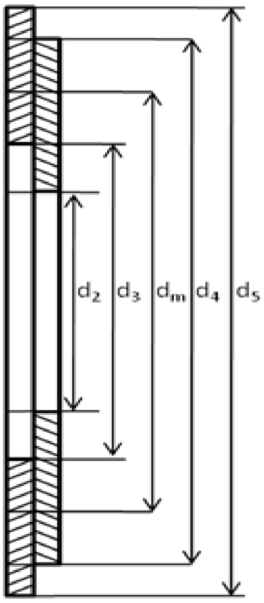


Figure A.2: Clutch lamellae pair

Table A.5: Choice of discs / lamellae pairs

	Disc Class = {I, II, III}
Thickness (mm)	b_1 = {2, 4, 6}
Chassis lamellae outer diameter (mm)	d_5 = {264, 336, 400}
Shaft lamellae outer diameter (mm)	d_4 = {195, 275, 354}
Chassis lamellae inner diameter (mm)	d_3 = {175, 220, 310}
Shaft lamellae inner diameter (mm)	d_2 = {153, 216, 300}
Medium diameter of friction surface (mm)	d_m = {185, 248, 332}
Moment of inertia of outer lamella (kgm²)	$J_{outer\_lamella}$ = {0.002, 0.01, 0.13}
Moment of inertia of inner lamella (kgm²)	$J_{inner\_lamella}$ = {0.0004, 0.003, 0.06}

The variables of the problem and their initial domains are listed and explained in Table A.6.

Table A.6: Variables of the multi-disc clutch design problem

	Remark	Initial Domain	Explanation
$relation_t\ (no\ unit)$	*continuous*	[1.3, 3]	Practical value, which defines the relation between torques
$l_1\ (mm)$	*continuous*	[1, 300]	Length of chassis
$l_2 (mm)$	*continuous*	[1, 300]	Length of chassis cylinder
$l_{total} (mm)$	*continuous*	[1, 600]	Length of total clutch
$b_3 (mm)$	*continuous*	[10, 300]	Width of shaft shoulder
$b_4 (mm)$	*continuous*	[1, 300]	Width of lamellae packet
$d_1 (mm)$	*continuous*	[0, 200]	Diameter of shaft drilling
$d_6 (mm)$	*continuous*	[200, 700]	Diameter of chassis
$i\ \ (no\ unit)$	*discrete*	[2, 18]	Amount of lamellae / friction pairs
$gear\_ration\ \ (no\ unit)$	*discrete*	[1, 32]	Ratio of gearbox, situated between engine and clutch
$m_{load}\ (kg)$	continuous	[100, 10000]	weight of load
$F_{clutch}\ (N)$	continuous	[1, 40000]	Force to pull the lamellae together
$t_{total}\ (sec)$	continuous	[0, 100]	Duration of whole clutching process and afterwards just until engine reaches initial speed again
$Z\ (hour^{-1})$	discrete	[1, 10000]	Clutching events per hour
$m_{clutch}\ (kg)$	continuous	[1, 2000]	Weight of dimensioned clutch
$m_{wholesystem}\ (kg)$	continuous	[0, 2000]	Weight of whole system (clutch + engine)
$l_{total}\ (mm)$	continuous	[0, 600]	Length of entire clutch
$d_{total}\ (mm)$	continuous	[0, 1000]	Diameter of entire clutch
$T_{load}\ (Nm)$	continuous	[0, 10000]	Torque on shaft caused by weight of load
$T_{engine_{red}}\ (Nm)$	continuous	[0, 10000]	Torque of the engine, reduced (after the gear box) on the clutch shaft
$t_{friction}\ (sec)$	continuous	[0.1, 20]	Duration of sliding of lamellae just after start of clutching
$J_{engineside}\ (kgm^2)$	continuous	[0, 100]	Moment of inertia of whole engine side
$J_{engine_{red}} (kgm^2)$	continuous	[0, 100]	Moment of inertia of engine, reduced on clutch shaft
$J_{clutch_{engineside}} (kgm^2)$	continuous	[0, 100]	Moment of inertia of chassis + outer lamellae
$J_{chassis} (kgm^2)$	continuous	[0, 100]	Moment of inertia of chassis
$J_{loadside} (kgm^2)$	continuous	[0, 1000]	Moment of inertia of whole load side
$J_{clutch_{loadside}}\ (kgm^2)$	continuous	[0, 100]	Moment of inertia of shaft + inner lamellae
$J_{shaft}\ (kgm^2)$	continuous	[0, 100]	Moment of inertia of shaft
$J_{weight_{red}}\ (kgm^2)$	continuous	[0, 1000]	Moment of inertia of load, reduced on shaft
Variables describing the clutching process			
$Q_{friction}\ (Nm)$	continuous	[100, 100000]	Heat energy, generated during the sliding
$T_{friction}\ (Nm)$	continuous	[0, 10000]	Friction torque between load side and engine side during clutching
$\omega_{engineside\_0}\ (sec^{-1})$	continuous	[0, 100]	Angular velocity of the chassis before clutching
$v_{relative\_0}\ \left(\frac{m}{s}\right)$	continuous	[0, 100]	Highest relative speed between inner and outer lamellae (at diameter d_4)
Variables for heat calculations			

$A_K(m^2)$	continuous	[0, 100]	Surface of clutch, exchanging heat with environment (chassis cylinder)
$\alpha_K\left(\frac{W}{m^2K}\right)$	continuous	[0, 100]	Coefficient: exchange of heat with environment
$v_K\left(\frac{m}{s}\right)$	continuous	[0, 1000]	Turning speed of cooling surface / chassis
Variables for stiffness and security calculations			
$\tau_{shaft\_shear}\ \left(\frac{N}{mm^2}\right)$	continuous	[0, 1200]	Shear stress caused by clutching force
$\sigma_{flex}\ \left(\frac{N}{mm^2}\right)$	continuous	[0, 1200]	Stress caused by torque due to clutching force
$\tau_{shaft\_torsion}\ \left(\frac{N}{mm^2}\right)$	continuous	[0, 1200]	Stress caused by torque on shaft caused by weight of load
$W_{polar\_shaft}\ (mm^2)$	continuous	[0, 1000000000]	Resistance of shaft against torque
$\sigma_{pulling\_force_{centrifugal\_shaft}}\ \left(\frac{N}{mm^2}\right)$	continuous	[0, 2000]	Stress in shaft caused by centrifugal forces
$\sigma_{pulling\_force_{centrifugal\_chassis}}\ \left(\frac{N}{mm^2}\right)$	continuous	[0, 2000]	Stress in chassis caused by centrifugal forces
$\tau_{chassis\_torsion}\ \left(\frac{N}{mm^2}\right)$	continuous	[0, 1200]	Stress caused by torque on chassis caused by weight of load
$W_{polar\_chassis}\ (mm^2)$	continuous	[0, 1000000000]	Resistance of chassis against torque
Safety Variables			
$Safety_1\ (no\ unit)$	continuous	[0, 300]	Safety against stress at position 1
$Safety_2\ (no\ unit)$	continuous	[0, 300]	Safety against stress at position 2
$Safety_3\ (no\ unit)$	continuous	[0, 300]	Safety against stress at position 3
$Safety_{pressure}\ (no\ unit)$	continuous	[0, 300]	Safety of discs material against pressure
Temperature Variables			
$temp_{final}\ (°C)$	continuous	[0, 400]	Final temperature of the clutch
$temp_{clutching}\ (°C)$	continuous	[0, 400]	Temperature during clutching, depending on amount of clutches per hour

The relations between the problem variables are defined through constraints in Equations (A.1-A.44).

The geometrical relations are shown by Equations (A.1-A.4):

$$l_1 = l_2 + c_2 \tag{A.1}$$

$$l_2 = c_1 + b_3 + b_4 \tag{A.2}$$

$$b_4 = b_1 i \tag{A.3}$$

$$l_{total} = l_1 + b_5 \tag{A.4}$$

The weight relations are shown by Equations (A.5-A.7).

The weight of the clutch is the sum of the weights of the chassis, the clutch disc and the shaft:

$$m_{clutch} = \rho_2 \cdot \pi \left(\left(\frac{d_5}{2} \right)^2 - \left(\frac{d_3}{2} \right)^2 \right) \cdot \frac{i \cdot b_1}{2} + \rho_2 \cdot \pi \left(\left(\frac{d_4}{2} \right)^2 - \left(\frac{d_2}{2} \right)^2 \right) \frac{i \cdot b_1}{2} + \rho_1 \cdot \pi \left(\frac{d_6}{2} \right)^2 (l_1 - l_2) + \rho_1 \cdot \pi \left(\left(\frac{d_6}{2} \right)^2 - \left(\frac{d_5}{2} \right)^2 \right) l_2 + \rho_1 \cdot \pi \left(\left(\frac{d_2}{2} \right)^2 - \left(\frac{d_1}{2} \right)^2 \right) (b_3 + b_4 + b_5) + \rho_1 \pi b_3 \left(\left(\frac{d_4}{2} \right)^2 - \left(\frac{d_2}{2} \right)^2 \right) \qquad \text{(A.5)}$$

The weight of the whole system is the sum of the weights of the engine and the the clutch:

$$m_{wholesystem} = m_{engine} + m_{clutch} \qquad \text{(A.6)}$$

The weight that has to be lifted implies the torque T$_{\text{load}}$:

$$\frac{T_{load}}{\left(\frac{d_{hoisting\_drum}}{2} \right) \cdot g} = m_{load} \qquad \text{(A.7)}$$

The clutching technology relations are shown by Equations (A.8-A.16).

The torque of the engine is augmented by the gear box:

$$T_{enginered} = T_{engine} \cdot gear\_ratio \qquad \text{(A.8)}$$

The friction torque T$_{\text{friction}}$ is derived by the size of the clutch, the amount of discs and the force which pulls these discs together:

$$T_{friction} = F_{clutch} \cdot friction\_value \cdot i \cdot \left(\frac{d_m}{2} \right) \qquad \text{(A.9)}$$

There are some relations between the engine torque, the torque created by the mass and the torsion torque. These need to be fulfilled in order to enable a successful clutching process:

$$T_{friction} = relation_T \cdot T_{load} \qquad \text{(A.10)}$$

$$T_{friction} = relation_T \cdot T_{engine\_red} \qquad \text{(A.11)}$$

$$T_{engine\_red} \geq \; 1.01 \; \cdot \; T_{load} \qquad \text{(A.12)}$$

The maximal angular velocity of the chassis can be calculated by the gear ratio and the rated speed of the engine. This speed reflects the relative angular velocity between the chassis and the shaft ($w_{shaft} = 0$) just before the clutching event:

$$w_{engineside\_0} = \frac{2 * \pi \cdot n_{engine\_0}}{gear\_ratio} \qquad \text{(A.13)}$$

The maximal relative speed between both surfaces is calculated by the maximal relative angular velocity and the outer diameter of the clutch discs:

$$v_{relative\_0} = \; w_{engineside\_0} \cdot \left(\frac{d_4}{2}\right) \qquad \text{(A.14)}$$

$$t_{friction} = \frac{\omega_{engineside\_0}}{\left(\frac{T_{friction} - T_{engine_{red}}}{J_{engineside}}\right) + \left(\frac{T_{friction} - T_{load}}{J_{loadside}}\right)} \qquad \text{(A.15)}$$

$$v_{relative0} \leq \; v_{relative\_max} \qquad \text{(A.16)}$$

The structural resistance relations are shown by Equations (A.17-A.28).

The force F_{clutch} leads to the shear stress $\tau_{shaft\ shear}$ and additionally to the flexural stress σ_{flex} at Position 1:

$$\tau_{shaft\_shear} = \frac{F_{clutch}}{d_2 \cdot \pi \cdot b_3} \qquad \text{(A.17)}$$

$$\sigma_{flex} = \frac{6 \cdot F_{clutch} \cdot \left(\frac{d_4 - d_2}{2}\right) \cdot \left(\frac{2}{3}\right)}{d_2 \cdot \pi \cdot {b_3}^2} \qquad \text{(A.18)}$$

An equivalent stress is derived from these stresses. It is compared with the allowed material stress in order to determine $Safety_1$:

$$Safety_1 = \frac{1}{\sigma_{max}} \sqrt{\sigma_{flex}^2 + 3 \cdot \tau_{shaft\_shear}^2} \qquad \text{(A.19)}$$

F_{clutch}, the friction surface and the maximum allowed material pressure are utilized in order to determine the safety of the clutch discs against pressure:

$$Safety_{pressure} = \; p_{max} * \; \frac{F_{clutch}}{\frac{{d_4}^2 - {d_3}^2}{4}} \qquad \text{(A.20)}$$

Due to the torque and centrifugal torque a stress is induced at Position 2. Thus, the thickness of the shaft has to be adapted. A high load weight creates a high torque:

$$\tau_{shaft\_torsion} = \frac{T_{load}}{W_{polar\_shaft}} \quad \text{(A.21)}$$

The thicker the shaft, the bigger is the resistive torque and the higher is the security:

$$W_{polar\_shaft} = \frac{\pi \cdot ({d_2}^4 - {d_1}^4)}{16 \cdot d_2} \quad \text{(A.22)}$$

The turning speed of the shaft creates a centrifugal force and stress:

$$\sigma_{pulling\_force_{centrifugal\_shaft}} = \left(\frac{d_2}{2}\right)^2 \cdot \omega^2_{engineside\_0} \cdot \rho_1 \quad \text{(A.23)}$$

An equivalent stress is derived from these stresses. It is compared with the allowed material stress in order to determine *Safety*$_2$:

$$Safety_2 = \frac{1}{\sigma_{max}} \sqrt{\sigma^2_{pulling\_force_{centrifugal\_shaft}} + 3 \cdot \tau^2_{shaft\_torsion}} \quad \text{(A.24)}$$

Due to the torque and centrifugal torque a stress is induced at Position 3 in the chassis. Thus, the thickness of the shaft has to be adapted. A high load weight creates a high torque:

$$\tau_{chassis\_torsion} = \frac{T_{load}}{W_{polar_{chassis}}} \quad \text{(A.25)}$$

The thicker the chassis, the bigger is the resistive torque and the higher is the security:

$$W_{polar\_chassis} = \frac{\pi \cdot ({d_6}^4 - {d_5}^4)}{16 \cdot d_6} \quad \text{(A.26)}$$

The turning speed of the chassis creates a centrifugal force and stress:

$$\sigma_{pulling\_force_{centrifugal\_chassis}} = \left(\frac{d_6}{2}\right)^2 \cdot \omega^2_{engineside\_0} \cdot \rho_1 \quad \text{(A.27)}$$

An equivalent stress is derived from these stresses. It is compared with the allowed material stress in order to determine *Safety₃*:

$$Safety_3 = \frac{1}{\sigma_{max}} \cdot \sqrt{\sigma^2_{pulling\_force_{centrifugal\_chassis}} + 3 \cdot \tau^2_{chassis\_torsion}} \quad \text{(A.28)}$$

The moment of inertia relations are shown by Equations (A.29-A.36). These are utilized to describe the dynamics of the clutching progress.

On the left side, there are the moments of inertia of the engine and of the clutch. Both have to be reduced on the clutch itself:

$$J_{engineside} = J_{engine_{red}} + J_{clutch_{engineside}} \quad \text{(A.29)}$$

The engine has to be reduced on the clutch shaft, as the clutching event is determined in the clutch itself:

$$J_{engine_{red}} = J_{engine} \cdot \, gear\_ratio^2 \quad \text{(A.30)}$$

The clutch itself has as well a moment of inertia which is composed of the chassis and the driving discs:

$$J_{clutch_{engineside}} = J_{chassis} + \frac{i}{2} * J_{outer\_lamella} \quad \text{(A.31)}$$

The moment of inertia for the chassis is derived from the outer diameter and the size constraints given by the chosen disc class:

$$J_{chassis} = \rho_1 \cdot \pi \left(\frac{d_6}{2}\right)^2 (l_1 - l_2) \frac{{d_6}^2}{4} + \rho_1 \cdot \pi \left(\left(\frac{d_6}{2}\right)^2 - \left(\frac{d_5}{2}\right)^2\right) l_2 \frac{{d_6}^2 + {d_5}^2}{4} \quad \text{(A.32)}$$

On the left side, there are the moments of inertia created by the load of the weight and the load of the clutch shaft. Both have to be reduced on the clutch itself:

$$J_{loadside} = J_{weight_{red}} + J_{clutch_{loadside}} \quad \text{(A.33)}$$

The weight has to be reduced on the shaft of the clutch:

$$J_{weight_{red}} = m_{load} \left(\frac{d_{hoisting\_drum}}{2}\right)^2 \quad \text{(A.34)}$$

The clutch itself has as well a moment of inertia. It is composed of the moment of inertia for the shaft and for the driven discs:

$$J_{clutch_{loadside}} = J_{shaft} + \frac{i}{2} \cdot J_{inner\_lamella} \qquad \text{(A.35)}$$

The moment of inertia for the shaft is derived from the inner bearing diameter and the size constraints given by the chosen disc class:

$$J_{shaft} = \rho_1 \cdot \pi \left(\left(\frac{d_2}{2}\right)^2 - \left(\frac{d_1}{2}\right)^2 \right) (b_3 + b_4 + b_5) \frac{{d_2}^2 + {d_1}^2}{4} + \rho_1 \cdot \pi \left(\left(\frac{d_4}{2}\right)^2 - \left(\frac{d_2}{2}\right)^2 \right) b_3 \frac{{d_4}^2 + {d_2}^2}{4} \qquad \text{(A.36)}$$

The heat and temperature relations are shown by Equations (A.37-A.44).

The heat energy depends on the friction torque between the discs, the velocity right at the beginning of the clutching event and its duration:

$$Q_{friction} = \frac{T_{friction}}{2} w_{engineside\_0} \cdot t_{friction} \qquad \text{(A.37)}$$

The maximum final temperature is the sum of environmental temperature and the temperature increase due to the various clutching events:

$$temp_{final} = temp_0 + temp_{clutching} \qquad \text{(A.38)}$$

The additional temperature increase is caused by the generated heat energy after each clutching event. The amount of this temperature can be decreased by the surface of the chassis which is turning and therefore cooling the components:

$$temp_{clutching} = \frac{Q_{friction} \cdot Z}{12.8 \cdot A_K} \qquad \text{(A.39)}$$

The cooling surface is determined by the outer diameter of the chassis and its length:

$$A_K = d_6 \cdot \pi \cdot l_1 \qquad \text{(A.40)}$$

Equations A.41 and A.42 describe the relations between the amount of clutching events per hour and the heat energy which is created by the clutching events (taking into account the material of the clutching discs (A.41) and the conductivity of the chassis material (A.42)).

$$Z = q_{max} \cdot i \cdot \frac{1}{Q_{friction}} \cdot \frac{{d_4}^2 - {d_3}^2}{4} \qquad \text{(A.41)}$$

$$Z = \frac{(temp_{max} - temp_0)(2\pi \cdot b_4 \cdot \lambda)}{Q_{friction} \cdot \log\left(\frac{d_6}{d_5}\right)} \qquad \text{(A.42)}$$

The final temperature after the clutching will be higher than before the clutching event, but should be lower than the maximum allowed temperature:

$$temp_{final} \geq temp_0 \qquad \text{(A.43)}$$

$$temp_{final} \leq temp_{max} \qquad \text{(A.44)}$$

Appendix B: Serious Game Transformation

The Serious Game transformation of the multi-disc clutch problem is presented below. This analogy represents clearly what should be improved and how it should be improved in the multi-disc clutch problem, without any redundant information.

Game Definitions and Rules

You are four lucky students. Your university has decided to offer you a scholarship for an internship abroad. However, the resources of your university for the scholarships are limited. This limit is confidential, not shared with you. You are going to play a game to share the resources for your monthly scholarship payments. In the game you are called "Agent". Before starting the game, define who are Agent 1, Agent 2, Agent 3 and Agent 4.

The payment limit is € 1,000 per month for each student (this does not mean that the resources of your university are € 4,000 per month in total. They are less, and you do not know how much it is!). You are going to live abroad, so you will receive the monthly pay in the local currency. The money transfer limit is 1,000 units of that currency per month for each student.

Each of you is going to define an imaginary country, and enter the exchange rate value of its local money currency against euro. However, you cannot go to a country where the local money currency is stronger than euro. Thus, this value can be between 0 and 1 (this is the compromise threshold value).

The monthly payment value is called WB, and it is equal to the amount of payment in local currency divided by 1000.

For example:

- You go to a country where the local money currency is X.
- Exchange rate: 1 X = € 0.5 (the threshold of compromise value is thus 0.5)
- If your WB = 1, it is equivalent of 0.5 x 1000 = 500 €.

In the game, you are going to negotiate with your school to increase your monthly purse. You start the game with 0.01 WB, and you increase it up to 1 WB (transfer limit). The game consists of iterations of negotiations, and each negotiation passes one after the other at iteration. The negotiator is chosen randomly. When it is your turn to negotiate; if you're not satisfied by the amount of your scholarship, you choose to negotiate by entering 1 on your computer screen. If you are currently satisfied by the amount of your scholarship you can pass your turn by entering 0.

If you want to negotiate, you need to define a constraint with a coefficient (Coefficient of Restriction: CR) between 0 and 9 in order to increase the scholarship amount:

- New WB value> = (1 + CR) x Old WB value
- If ((1 + CR) x Old WB value) exceeds 1, the constraint is set automatically:
    - New WB value = 1. This is because the money transfer limit is 1 WB.
- Since the CR is always between 0 and 9, and you start the game with 0.01 WB, a request of WB=1 is prevented at the first iterations of the game.

If the resources of your school are not sufficient for the payment you ask, your constraint will be rejected. You are thus (potentially) in conflict with the other students:

- Either your conflict is justified, because your WB amount is not the highest among the four students. Your payment increase is potentially blocked by another student (limited resource).
- Either your conflict is not justified because your WB is the highest among the four students.

The conflict justification is performed automatically by the program.

When four students are processed at iteration, the process passes to the next iteration. You can exit the game if you are completely satisfied with your monthly payment. In order to exit the game you need to type a CR equal to 0. Also, you exit the game automatically if your WB is equal to 1. The game is over when all the students are out of the game.

There are four types of game. Each game is driven by a different strategy. You are going to play with one after the other one. Some strategies are able to resolve conflicts. The conflict is resolved by another student who can help you by decreasing his/her WB. Your school thus has enough resource for your request and your conflict is resolved. The student that is able to help is detected and the conflict is resolved automatically by the program. The game strategies are as follows:

- Strategy 1: You cannot communicate with the other students and look at their screens during the game. You cannot ask for help or attempt to resolve a conflict.

- Strategy 2: You share your screens with the other students. You can ask for help from other students that can also request your help. Once the program detects you as the person who can help, it is your desire to help or not to help.

- Strategy 3: You share your screens with the other students. You can ask for help from other students that can also request your help. It is your desire to help or not to help. If you choose not to help, you will however be penalized: you will be blocked at next iteration of the process.

- Strategy 4: You share your screens with the other students. You can ask for help from other students that can also request your help. Help that is requested cannot be refused.

Remember that your act is selfish:

- If you go to an expensive country in which the currency exchange value is higher

- If you define constraints frequently

- If your CR is larger

- If you do not agree to help

Do not forget that the other students are your close friends. If you are very selfish, your friends will suffer in a foreign country. Additionally, if a student’s WB is very low, the university administration can decide to cancel the scholarship program next year, since they do not accept their students to suffer.

You can be whatever you want in a game; selfish or selfless. However, reflect the same personality for each game strategy. For instance; if you are selfish in the game of Strategy 1, then be selfish in the others. Select the same agent and keep the same currency exchange value for each game.

Abstract

In the product dimensioning phase of a distributed design, inconsistencies can emerge among design objectives as well as among working procedures of heterogeneous subsystems. In this phase, design actors which compose subsystems must collaborate concurrently, since their works are linked to each other through dimensioning couplings among their sub-problems. Inconsistencies through these couplings yield thus to design conflicts. The issue is how to obtain a collaborative convergence to satisfy the global and individual objectives of design actors when making design decisions under uncertainty. The objective of this dissertation is to propose a model for preventing and resolving design conflicts in order to obtain a collaborative convergence, while overcoming the design uncertainty through Set-based Design (SBD). Design attitudes are modeled with Belief-Desire-Intention paradigm to explore inconsistencies and manage conflicts in design processes. The conventional bottom-up approach is thus extended through agent-based attitude modeling techniques. In this approach, design agents can set requirements directly on their wellbeing values that represent how their design targets are likely to be met at a given moment of the design process. Monte Carlo simulations are performed to evaluate the performance of this approach, providing a variety of agent attitudes. Compared to conventional bottom-up and top-down design approaches, the results reveal a fewer number of design conflicts and a reduced aggregated conflict intensity. Constraint satisfaction problem (CSP) techniques and design attitudes are both applied to detect and justify design conflicts of heterogeneous design agents. A novel cooperative CSP (CoCSP) is developed in order to resolve design conflicts through compromising constraint restriction. The conflict resolution system can be adopted for different proposed strategies which take into account the solidarity architecture of design agents. The simulation results show that while promoting solidarity in distributed design by helping agents that suffer, the conflict intensity is reduced, and better design results are obtained.

Keywords: Distributed design, Collaborative design, Concurrent engineering, Conflict prevention, Conflict resolution, Set-based design, Constraint satisfaction problem, Multi-agent systems

Résumé

En conception distribuée, dans la phase du dimensionnement du produit, des incohérences peuvent émerger entre les objectifs de conception et entre les procédures de travail des sous-systèmes hétérogènes. Dans cette phase, les acteurs de conception doivent collaborer d'une manière concourante, car leurs tâches sont reliées les unes aux autres par les couplages de dimensionnement entre leurs sous-problèmes. Les incohérences peuvent provoquer des conflits de conception en raison de ces couplages. La question est de savoir comment obtenir une convergence collaborative pour satisfaire les objectifs globaux et individuels des acteurs de conception lorsque ces acteurs prennent des décisions de conception sous incertitude. L'objectif de cette thèse est de proposer un modèle pour empêcher et résoudre les conflits de conception, tout en surmontant le problème de l'incertitude de la conception avec l'approche de « conception basée sur les ensembles » (SBD). Pour cela, les attitudes de conception sont modélisées avec le paradigme « Croyances-Désirs-Intentions » afin d'explorer les incohérences et gérer les conflits dans les processus de conception. L'approche ascendante conventionnelle est ainsi étendue grâce à des techniques de modélisation multi-agents. Dans cette approche, les agents de conception peuvent fixer des exigences directement sur leurs indicateurs de « bien-être ». Ces indicateurs représentent la manière dont leurs objectifs de conception sont susceptibles d'être satisfaits à un moment donné du processus. Des simulations de Monte Carlo sont effectuées pour évaluer la performance de cette approche, offrant une variété d'attitudes de l'agent. Par rapport aux approches classiques de conception ascendante et descendante, les résultats révèlent moins de conflits de conception et une intensité des conflits réduite. Les techniques de « problème de satisfaction de contraintes » (CSP) et les attitudes de conception sont appliquées pour détecter et justifier des conflits de conception entre les agents hétérogènes. Une nouvelle forme du modèle « Cooperative CSP » (CoCSP) est ainsi mise au point afin de résoudre les conflits de conception en détectant le compromis entre les contraintes. Le système de résolution des conflits peut être adopté grâce à différentes stratégies proposées qui prennent en compte l'architecture de solidarité des agents. Les résultats des simulations montrent que l'intensité des conflits en conception distribuée est réduite par la promotion de la solidarité qui déclenche une aide aux agents en souffrance.

Mots-clés: Conception distribuée, Conception collaborative, Ingénierie concourante, Evitement des conflits, Résolution des conflits, Conception basée sur les ensembles, Problème de satisfaction de contraintes, Systèmes multi-agents