Resolving design conflicts and evaluating solidarity in distributed design

Baris Canbaz, Bernard Yannou, Pierre-Alain Yvars

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

Resolving design conflicts and evaluating solidarity in distributed design

Baris Canbaz, Bernard Yannou, Pierre-Alain Yvars

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 in order 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 with 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.

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. 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 [1]. In distributed design, while the global design problem is decomposed into sub-problems, design responsibility is decentralized and distributed to organizational subsystems composed of one or more design experts [2]. 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 [3] point out, it is highly unlikely to obtain true concurrency in reality, because subsystems are not independent. They 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 [4]. For instance, a designer that influences the design model more frequently and restrictively can block the other designers which are 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 [5] proposes a heuristic-based computational model that produces advice for resolving conflicts between design agents. The model utilizes the knowledge about conflict resolution strategies obtained from empirical design expertise. Wong [6] proposes a method of cooperative knowledge-based systems that includes a library of multi-agent design conflict resolution strategies. These strategies 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 [7] define a constraint-based dynamic design system model that includes facilitator agents. They send messages to relax some

constraints until a consistent solution is obtained. Li et al. [8] 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. [9] 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. [10] propose a combinatorial heuristic algorithm for design conflict negotiation which is based on Fuzzy Matter Element Particle Swarm Optimization (FMEPSO). Jin et al. [11] 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. [12] 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. [13].

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 [14]. According to Malak et al. [15], 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 [16], we defined a novel bottom-up design approach that employs the concept of Set-based Design (SBD) for managing imprecision in design. Design agent attitudes were modeled 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 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 3, and the CSP simulation process of this model is presented in Section 4. Monte Carlo simulations of different conflict resolution systems and the non-cooperative design system are performed on a design problem of a multi-clutch system that involves variable agent characters. In Section 5, we present the simulation problem definitions and simulation results. Different system strategies are compared with regard to their gains and costs.

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 [17], [18]. 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 [19], [20]. 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 [21], as well as robustness to design [22]. 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. [23]–[26]. 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 [27]. A multidimensional space is defined by the Cartesian product of variable domains. It 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 [28]. For example, X and Y are integer variables, and $Z = X \times Y$. Their domains are $(X) = [20\,,30]$ and $D(Y) = [15\,,25]$; so the domain of Z is synthesized as $D(Z) = [300\,,750]$. If a constraint is defined as $X < Y$, 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 [28] 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 [29]. The problem evolves over time with some agent actions which are related to the constraints performed through a process. 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 Δ: $P^{t-1} \rightarrow P^t$ 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: $P = \left(P_{A_i} + \cdots + P_{A_n}\right)$ [30]. 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 [31]. This can be an advantage for large but not very densely coupled problems. However, as Salido and Barber [32] 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 [33], [34] 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. Since 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.

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 [35]. The most widely deployed architecture of an agent is the Belief-Desire-Intention (BDI) paradigm developed by Bratman et al. [36]. 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 [37]. We use CSP definitions and the BDI paradigm to manage design conflicts of autonomous agents. Fig. 1 shows our adaptation of the BDI mechanism for a design agent.

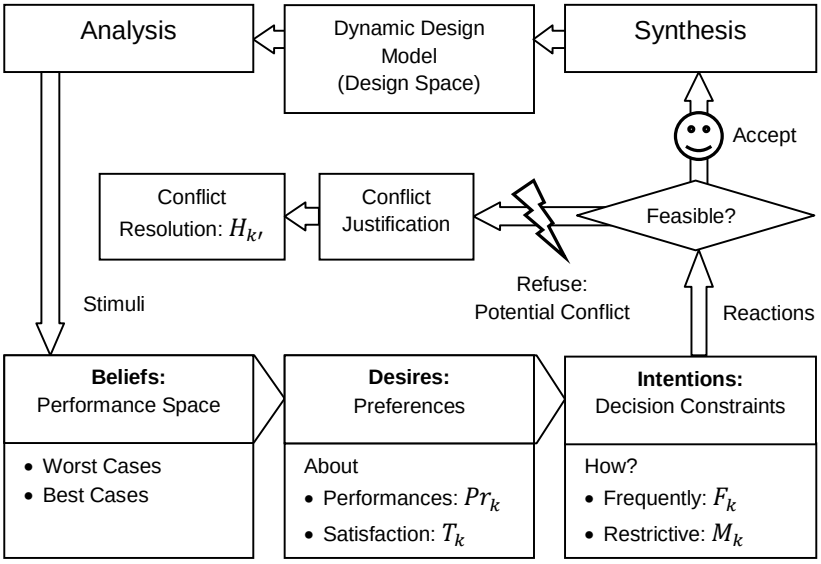

Fig. 1. BDI Mechanism of Design Agents

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 this 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 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. It 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. It represents 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. 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.

In our BDI adaptation, beliefs and intentions are dynamic, while desires are static. Preferences do not change during the design process, because we assume that they are defined at the initial process stage. With the increasing number of decision constraints introduced to the problem, the sub-problem of an agent is dynamic through the design process stages. The aggregated sub-problems propagate the dynamic design model that converges to a narrower space continuously. Therefore, beliefs evolve while the design uncertainty is reduced progressively. Further decisions are made when more detailed design information is available. Consequently, intentions change during the design process. Since design agents are related through couplings, their interactions are dynamic. The consistency of a decision constraint defined by an agent depends on the solution space which is also restricted by the other agents. A potential design conflict caused by an inconsistent constraint, either resolved or unresolved, will alter the intentions of agents at subsequent process stages.

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 and 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*. v_i is the performance value of the performance *i*. Then the piecewise constraints are as follows:

If $v_i \geq P_1$, $s_{ki} = 1$ (1)
If $v_i \leq P_2$, $s_{ki} = 0$ (2)
If $P_1 > v_i > P_2$, $1 > s_{ki} > 0$ (3)

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 $[x_i^t , y_i^t]$. Fig. 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.

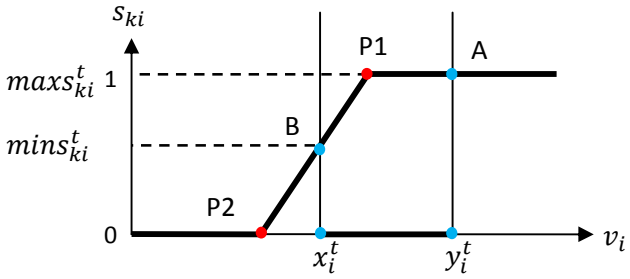

Fig. 2. Intervals of satisfaction function

A design agent may evaluate several design performances. The most widely employed decision making model in MAS for evaluating multiple performances is the multiple attribute utility theory (MAUT). In order to elicit the aggregate utility, weights of multiple attributes are estimated methodologically [38], [39]. The weights for design performances can be estimated if the sub-problem of the agent is scalable. Otherwise, the sub-problem should further be decomposed. In this paper, we assume that the design problem is correctly decomposed, so that an agent is able to assign weights to its design performances by comparing their relative importance for its job. 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:

$$mins_k^t = \sum_{i=1}^{I} w_{ki} \times mins_{ki}^t \quad (4)$$

$$maxs_k^t = \sum_{i=1}^{I} w_{ki} \times maxs_{ki}^t \quad (5)$$

$$\sum_{i=1}^{I} w_{ki} = 1 \ \forall k \qquad (6)$$

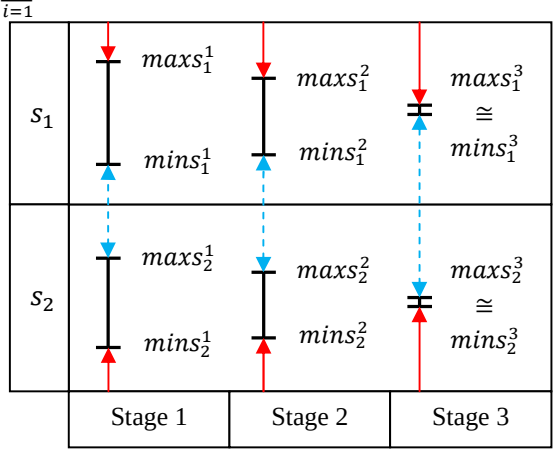

Fig. 3. Bilateral convergence

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. Fig. 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. However, this decreases the maximum satisfaction of the other agent with a conflicting objective. 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 of agent *k* about this satisfaction value is called compromise threshold T_k. This is the compromise attitude of agent *k*. 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. The larger the T_k value, the more restrictive agent *k* is. T_k is defined empirically by considering both technical and arbitrary factors: solution space, uncertainty, and the other agents' design freedom. It should be noted that, if agent *k* is very egoistic, it will not consider the other agents' design freedom. It can thus define a large T_k value.

Satisfaction values are normalized by dividing them by the compromise threshold value, and this provides wellbeing states Eq. (7, 8). Wellbeing is defined for each agent *k* 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 at stage *t*. As described in our earlier work [16], these indicators represent how agents' design targets are likely to be met at a given moment of the design process. Global states of design agents are thus observed. This is the reason why the "wellbeing" term is chosen to name these indicators. Since the convergence is bilateral, the likelihood to meet the design targets of an agent can be different than another one. Minimum wellbeing indicators of agents are compared to justify a potential conflict. If an agent suffers because of design conflicts, it will thus be detected immediately. Agent *k* defines decision constraints when it is not in the compromise state. The decision constraint of agent *k* is rejected if it does not provide any consistent solution. This represents a potential design conflict. 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 has 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.

$$minwb_k^t = \frac{mins_k^t}{T_k} \quad (7)$$

$$maxwb_k^t = \frac{maxs_k^t}{T_k} \quad (8)$$

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 $C_k^t: wb_k \geq minwb_k^t \times (1 + M_k)$ 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.

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. Fig. 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, 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.

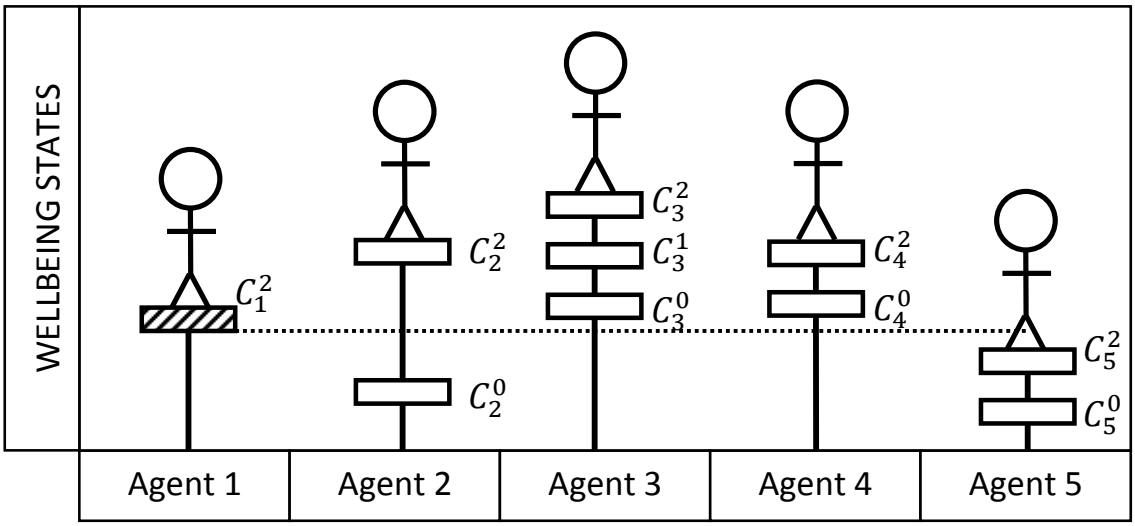

Fig. 4. Negotiation phase of design agents

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. Fig. 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: (C_1^1, C_1^3, C_1^5), L_2: $(C_2^0, C_2^1, C_2^2, C_2^3, C_2^4, C_2^5)$ and L_3: (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		$C_2^4: wb_2 \geq 0.85$	$C_3^4: wb_3 \geq 47$
t=5	$C_1^5: wb_1 \geq 1$	$C_2^5: wb_2 \geq 1$	
t=6			$C_3^6: wb_3 \geq 0.65$

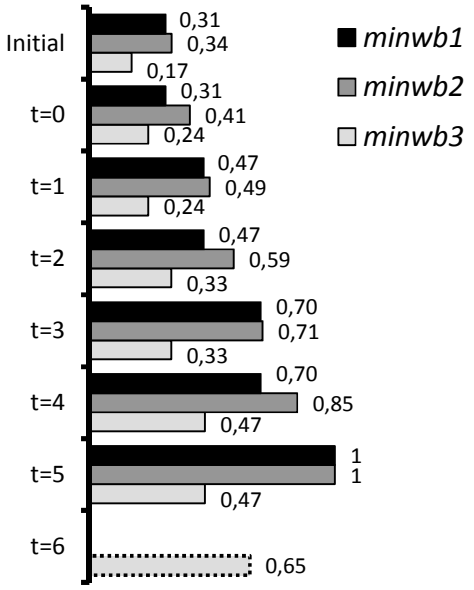

Fig. 5. Numerical example of conflict management

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. It sends another message to the agent that can help about exactly how the agent 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 Fig. 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. In addition, 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. 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: Non-cooperative 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.

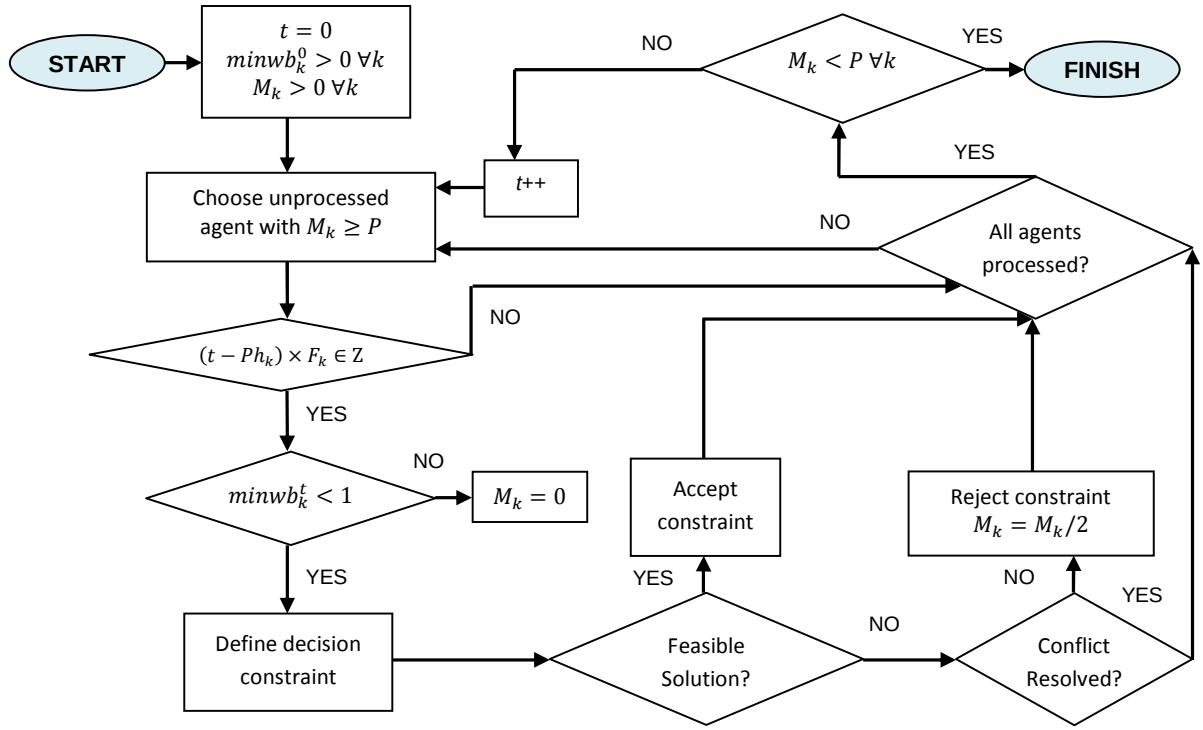

Fig. 6. Simulation algorithm

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 [40]. 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 Fig. 6. 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 whereas 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 wellbeing and divergence of individual solutions. 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 wellbeing values of agents while minimizing their divergence. This divergence is defined as the difference between agents' individual wellbeing states. It represents the degree of intensity of the unresolved design conflicts. In the ideal case, agents should obtain the same wellbeing values and each one should be equal to 1. Absolute differences of the wellbeing values of each two element combination represent a vector $D(d_1, \dots, d_n)$. The Euclidian distance of this vector solution to the ideal case solution gives the divergence of the individual solutions: $Divergence = \sqrt{(d_1)^2 + \dots + (d_n)^2}$. More divergent solutions lead to more intense conflicts, because the divergence is caused by agents with a relatively low wellbeing value. However, the divergence cannot be evaluated alone. It should be evaluated with the total wellbeing value because a zero divergence is not desirable if total wellbeing is zero.

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 [41]. 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 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 [42] 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 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 [43]. As shown in Fig. 7 it is a design problem of a multi-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 nomenclature is given in Table 2. The problem is distributed to four design agents. The design objectives are shown in Table 3. The piecewise constraints representing the preferences of the design agents are shown in Table 4. All the transitions between the preferences are considered linear as shown in Fig. 2. Agent 4 evaluates four design performances, the same weight being attributed to these performances: $s_4 = \sum_{i=1}^{4}(s_{4i} \times 0.25)$.

Table 2. Problem nomenclature

m_{load}	Weight of mass to be lifted, kg.
m_s	Weight of whole system (clutch + engine), kg.
T	Final temperature of the clutch, °C.
S_1	Safety against stress at position 1
S_2	Safety against stress at position 2
S_3	Safety against stress at position 3
S_p	Safety of discs material against pressure
s_k	Satisfaction of aent *k*
s_{4i}	Satisfaction of Agent 4 from performance *i*

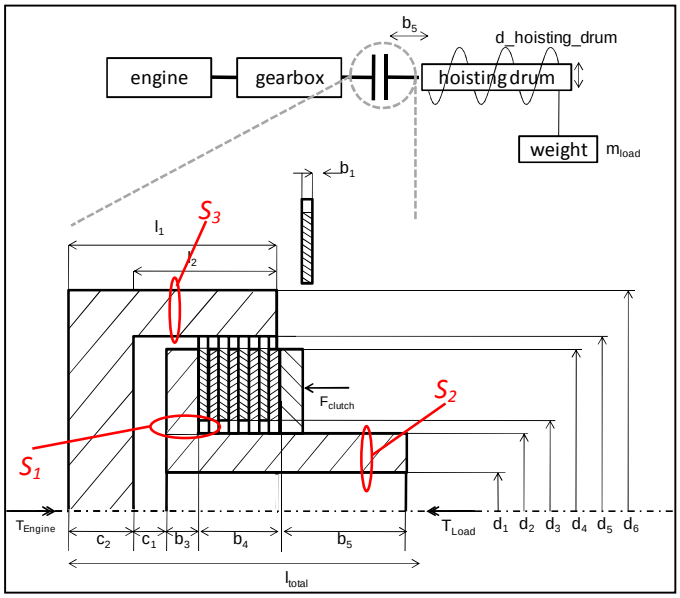

Fig. 7. Multi-clutch system

Table 3. Problem objectives

	Agent 1	Agent 2	Agent 3	Agent 4
Objectives				

Table 4. Clutch preferences

If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then
If	then

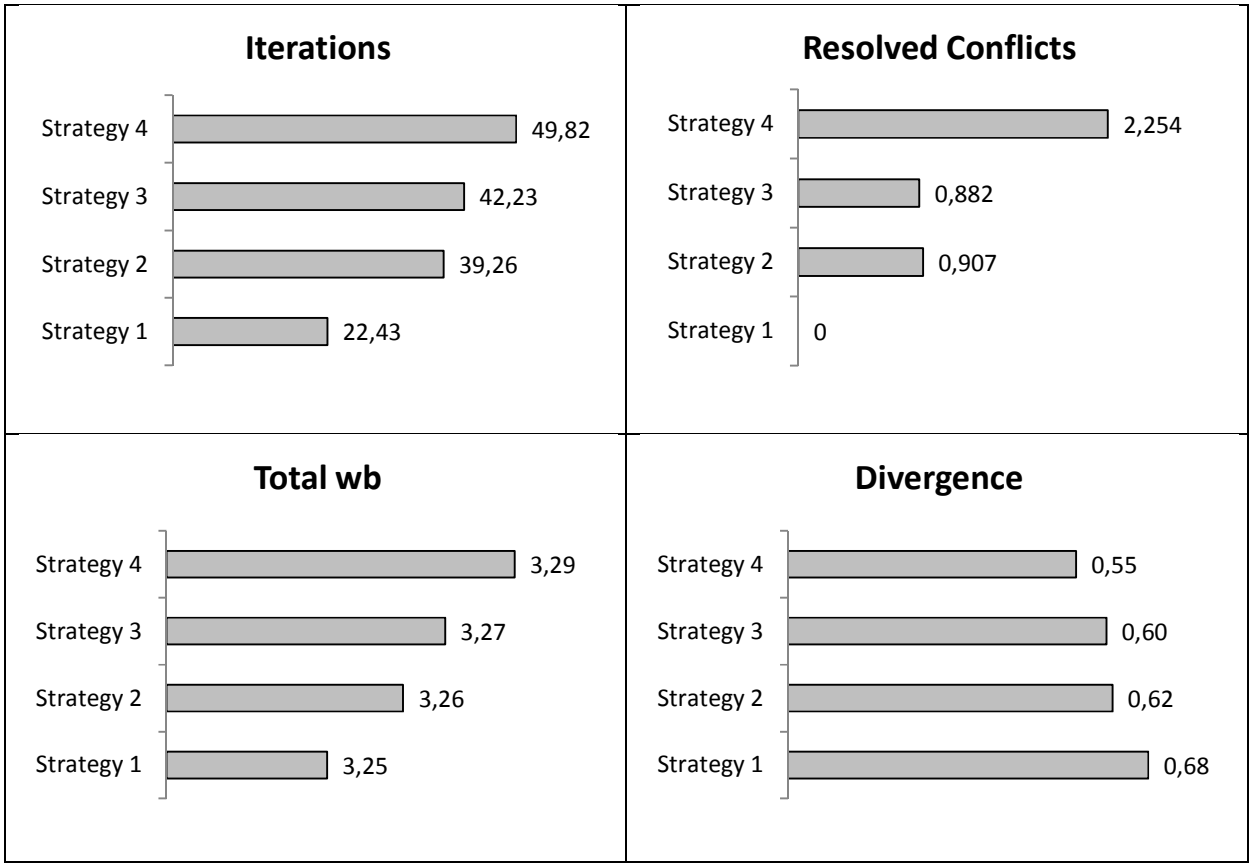

Fig. 8. Simulation Results

The average results of the Monte Carlo simulation are shown in Fig. 8. In order to analyze the statistical significance of the results, we performed two tailed t-tests for each pair of strategies. The resulting p-values are shown in Table 5. If the significance level is considered as 0.05, the results of the strategies are significantly different (very low p-values), except Strategies 2 and 3 (p-values larger than 0.05).

Table 5. p-values

	Strategies 1-4	Strategies 2-4	Strategies 3-4	Strategies 1-3	Strategies 2-3	Strategies 1-2
Iterations	1.15E-15	0.000855	0.026087	4.08E-17	0.186749	5.21E-17
Resolved Conflicts	5.59E-40	4.06E-18	7.29E-18	3.75E-87	0.488849	1.3E-91
Total wb	1.74E-06	0.000391	0.036001	0.001059	0.193819	0.0491
Divergence	6.25E-38	4.15E-15	2.96E-10	2.24E-16	0.180018	1E-12

In the ideal case, all four agents would obtain a final wellbeing value which is equal to 1. The total wellbeing would be equal to 4, and the divergence would be equal to 0. This ideal solution represents that all the desires of agents are fulfilled. However, this is a utopia solution, because the convergence of the wellbeing is typically bilateral. Since the desires of design agents reflect customer preferences, approaching the ideal solution along both the total wellbeing and divergence axes will improve the designed product. This will increase the overall utility of the product while

providing more balance between the satisfactions of different design objectives. Thus, the design process results in a balanced and optimal solution that best meets all relative customer preferences. As seen in Fig. 8, the conflict resolution systems, regardless of their adoption strategy, result in a larger total wellbeing value and a smaller divergence than the non-cooperative design system. This shows that the conflict resolution systems approach the ideal solution more closely than the non-cooperative design system. However, this is obtained at a cost of longer process time. The most rapid system strategy is the non-cooperative 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 wellbeing is increased with the number of design conflicts resolved or prevented. The least divergence and the greatest total wellbeing values are obtained by the centralized conflict resolution system. These results (total wb = 3.29 and divergence = 0.55) are the closest to the ideal solution (total wb = 4 and divergence = 0). Although the average results of Strategy 3 (total wb = 3.27 and divergence = 0.60) are slightly better than Strategy 2 (total wb = 3.26 and divergence = 0.62), it is not statistically significant that the controlled conflict resolution system outperforms the decentralized conflict resolution system.

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. The degree of this helping attitude represents the solidarity of agents. We defined three different conflict resolution system strategies by considering the solidarity architecture of the agents. We compared these strategies with the non-cooperative 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 non-cooperative design system. The divergence of individual wellbeing solutions is lowered and the total wellbeing 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. In addition, it should be noted that the proposed conflict management model can be applied on only measurable design systems where all the design aspects can be quantified. Another limitation is that the feasibility of the conflict resolution is not guaranteed. During a design process, the model may not detect a feasible help although design agents would help. This feasibility behavior depends on the previous accepted constraints.

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 wellbeing, this system converges to the ideal solution closer than any other system. This system thus produces a better final product which has a higher overall utility and a balanced satisfaction among different attributes. In addition, it is shown that the penalization of uncooperative agents do not significantly improve the solution in a conflict resolution system where the solidarity is autonomous. We conclude that the centralization of the conflict resolution system (solidarity obligation) is more fruitful and should be preferred to the decentralization (autonomous solidarity). 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.

References

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

[2] P. Y. Papalambros, N. F. Michelena, and N. Kikuchi, "Distributed Cooperative Systems Design," in *Proceedings of the 11th International Conference on Engineering Design*, Tampere, Finland, 1997, vol. 2, pp. 265–270.

[3] K. Lewis and F. Mistree, "Collaborative, Sequential, and Isolated Decisions in Design," *Journal of Mechanical Design*, vol. 120, no. 4, p. 643, 1998.

[4] L. Zhao and Y. Jin, "Work Structure Based Collaborative Engineering Design," in *Proceedings of ASME Design Engineering Technical Conferences*, 2003, pp. 865–874.

[5] M. Klein, "Supporting conflict resolution in cooperative design systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 6, pp. 1379–1390, 1991.

[6] S. T. C. Wong, "Coping with conflict in cooperative knowledge-based systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 27, no. 1, pp. 57–72, 1997.

[7] A. S. Koulinitch and L. B. Sheremetov, "Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor," *Expert Systems with Applications*, vol. 15, no. 3–4, pp. 295–307, Oct. 1998.

[8] X. Li, X. Zhou, and X. Ruan, "Conflict management in closely coupled collaborative design system," *International Journal of Computer Integrated Manufacturing*, vol. 15, no. 4, pp. 345–352, 2002.

[9] M. Shin, Y. Cha, K. Ryu, and M. Jung, "Conflict detection and resolution for goal formation in the fractal manufacturing system," *International Journal of Production Research*, vol. 44, no. 3, pp. 447–465, 2006.

[10] Y. Yin, L. Sun, and C. Guo, "A policy of conflict negotiation based on fuzzy matter element particle swarm optimization in distributed collaborative creative design," *Computer-Aided Design*, vol. 40, no. 10–11, pp. 1009–1014, Oct. 2008.

[11] G. Jin, B. Ying, and Z. Rong, "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*, 2009, pp. 1669–1674.

[12] K. W. Li, K. W. Hipel, D. M. Kilgour, and L. Fang, "Preference uncertainty in the graph model for conflict resolution," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 34, no. 4, pp. 507–520, 2004.

[13] M. A. Bashar, D. M. Kilgour, and K. W. Hipel, "Fuzzy Preferences in the Graph Model for Conflict Resolution," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 4, pp. 760–770, 2012.

[14] G. W. Parry, "The characterization of uncertainty in Probabilistic Risk Assessments of complex systems," *Reliability Engineering & System Safety*, vol. 54, no. 2–3, pp. 119–126, Nov. 1996.

[15] R. J. Malak, J. M. Aughenbaugh, and C. J. J. Paredis, "Multi-attribute utility analysis in set-based conceptual design," *Computer-Aided Design*, vol. 41, no. 3, pp. 214–227, Mar. 2009.

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

[17] E. K. Antonsson and K. N. Otto, "Imprecision in Engineering Design," *Journal of Mechanical Design*, vol. 117, no. B, pp. 25–32, 1995.

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

[19] A. C. Ward, J. Liker, D. K. Sobek, and J. J. Cristiano, "Set-based concurrent engineering and Toyota," in *Proceedings of ASME Design Engineering Technical Conferences*, 1994, vol. 68, pp. 79–90.

[20] D. K. Sobek, A. C. Ward, and J. Liker, "Toyota's Principles of Set-Based Concurrent Engineering," *Sloan Management Review*, vol. 40, no. 2, pp. 67–83, 1999.

[21] T. A. McKenney, L. F. Kemink, and D. J. Singer, "Adapting to Changes in Design Requirements Using Set-Based Design," *Naval Engineers Journal*, vol. 123, no. 3, pp. 66–77, 2011.

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

[23] B. Yannou, P.-A. Yvars, C. Hoyle, and W. Chen, "Set-based design by simulation of usage scenario coverage," *Journal of Engineering Design*, vol. 24, no. 8, pp. 575–603, 2013.

[24] Y. Meyer and P.-A. Yvars, "Optimization of a passive structure for active vibration isolation: an interval-computation- and constraint-propagation-based approach," *Engineering Optimization*, vol. 44, no. 12, pp. 1463–1489, 2012.

[25] J. H. Panchal, M. G. Fernández, C. J. J. Paredis, J. K. Allen, and F. Mistree, "An Interval-based Constraint Satisfaction (IBCS) Method for Decentralized, Collaborative Multifunctional Design," *Concurrent Engineering*, vol. 15, no. 3, pp. 309–323, Sep. 2007.

[26] B. Yannou and G. Harmel, "Use of Constraint Programming for Design," in *Advances in Design*, H. A. ElMaraghy and W. H. ElMaraghy, Eds. Springer London, 2006, pp. 145–157.

[27] U. Montanari, "Networks of constraints: Fundamental properties and applications to picture processing," *Information Sciences*, vol. 7, pp. 95–132, 1974.

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

[29] R. Dechter and A. Dechter, "Belief Maintenance in Dynamic Constraint Networks," in *American Association for Artificial Intelligence*, 1988, pp. 37–42.

[30] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara, "The distributed constraint satisfaction problem: formalization and algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 5, pp. 673–685, 1998.

[31] B. Faltings and M. Yokoo, "Introduction: Special Issue on Distributed Constraint Satisfaction," *Artificial Intelligence*, vol. 161, no. 1–2, pp. 1–5, Jan. 2005.

[32] M. A. Salido and F. Barber, "Distributed CSPs by graph partitioning," *Applied Mathematics and Computation*, vol. 183, no. 1, pp. 491–498, Dec. 2006.

[33] P.-A. Yvars, "A CSP approach for the network of product lifecycle constraints consistency in a collaborative design context," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 6, pp. 961–970, Sep. 2009.

[34] P.-A. Yvars, "A Constraint Based Decision Support System for Deadlocks Resolution in Collaborative New Product Design," *Journal of Decision Systems*, vol. 19, no. 1, pp. 57–74, 2010.

[35] M. Goyal, "Attitude based teams in a hostile dynamic world," *Knowledge-Based Systems*, vol. 18, no. 6, pp. 245–255, Oct. 2005.

[36] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Computational Intelligence*, vol. 4, no. 3, pp. 349–355, 1988.

[37] C. Castelfranchi, F. D. Rosis, R. Falcone, and S. Pizzutilo, "Personality Traits and Social Attitudes in Multiagent Cooperation," *Applied Artificial Intelligence*, vol. 12, no. 7–8, pp. 649–675, 1998.

[38] Y. Guo, J. P. Müller, and C. Weinhardt, "Learning User Preferences for Multi-attribute Negotiation: An Evolutionary Approach," in *Multi-Agent Systems and Applications III*, V. Mařík, M. Pěchouček, and J. Müller, Eds. Springer Berlin Heidelberg, 2003, pp. 303–313.

[39] C. M. Jonker, V. Robu, and J. Treur, "An agent architecture for multi-attribute negotiation using incomplete preference information," *Auton Agent Multi-Agent Syst*, vol. 15, no. 2, pp. 221–252, Oct. 2007.

[40] L. Granvilliers, "Adaptive Bisection of Numerical CSPs," in *Principles and Practice of Constraint Programming*, M. Milano, Ed. Springer Berlin Heidelberg, 2012, pp. 290–298.

[41] M. S. Pita and F. B. Lima Neto, "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*, New York, NY, USA, 2007, pp. 2927–2932.

[42] IBM, "IBM ILOG CPLEX CP Optimizer for Constraint Programs - Features and benefits," 12-Jun-2012. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/about/.

[43] B. Yannou, C. Mazur, and P.-A. Yvars, "Parameterization and Dimensioning of the Multi-Disc Clutch in the CO4 Environment," Ecole Centrale Paris, Technical report No. 2010-21, 2010. [Online]. Available: http://www.lgi.ecp.fr/Biblio/PDF/CR-LGI-2010-21.pdf