Improving Process Performance of Distributed Set-Based Design Systems by Controlling Wellbeing Indicators of Design Actors

Baris Canbaz, Bernard Yannou, Pierre-Alain Yvars

Canbaz B., Yannou B., Yvars P.-A., (2013), *Improving Process Performance of Distributed Set-based Design Systems by Controlling Wellbeing Indicators of Design Actors.* Journal of Mechanical Design.

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

Baris Canbaz
Ecole Centrale Paris
Laboratoire Genie Industriel
Chatenay-Malabry, France
baris.canbaz@ecp.fr

Bernard Yannou
Ecole Centrale Paris
Laboratoire Genie Industriel
Chatenay-Malabry, France
bernard.yannou@ecp.fr

Pierre-Alain Yvars
Institut Supérieur de Mécanique
de Paris (SupMeca) – LISMMA
3 rue Fernand Hainaut, 93407,
Saint Ouen Cedex, France
payvars@supmeca.fr

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.

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 [1]. 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. [2] 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 [3–5].

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 [6], 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 [7].

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 [8], 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. [9] and design process performance as shown by Park et al. [10]. 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 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 2 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 [11]. As represented in Fig.1, the intervals shrink with specifications shown with dashed arrows, and converge to a point solution S^C: $\left(s_1^C, s_2^C, s_3^C\right)$. 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 Fig. 1, a collaborative compromise solution S^C: $\left(s_1^C = 0.5,\ s_2^C = 0.5,\ s_3^C = 0.5\right)$ can be achieved with $a = 3$ and $b = 15$. Individual satisfactions are equal, so this solution has a zero divergence.

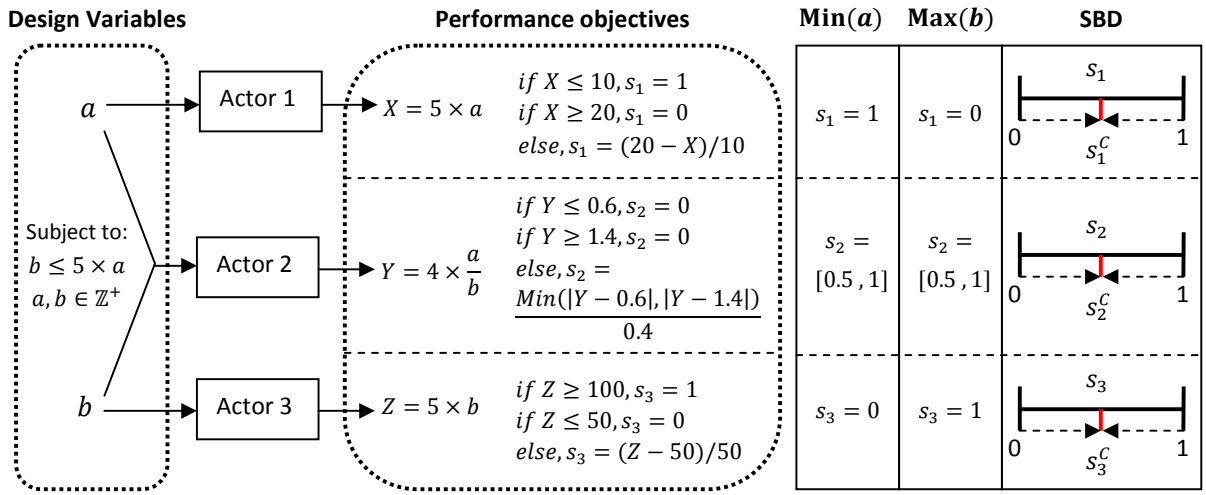

Figure 1. Coupled design pattern

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 [12]. 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 Sec. 2 the uncertainty management of SBD and Constraint Satisfaction Problem (CSP) techniques are discussed. Our distributed SBD framework is introduced in Sec. 3 and the CSP simulation process of this framework is presented in Sec. 4. Monte Carlo simulations of our approach are performed on a design problem. Problem definitions and simulation results are presented in Sec. 5. The results from Sec. 6 are discussed and further works are identified in Sec. 7.

2 Collaborative Design Approaches and SBD

Uncertainty representation and propagation are important issues for design quality in preliminary design [13]. 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. [14], 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 [15,16]. 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. This is shown in Fig. 2 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 lines represent the convergence.

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. [17] and Wang and Terpenny [18] 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. [14] 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 [16].

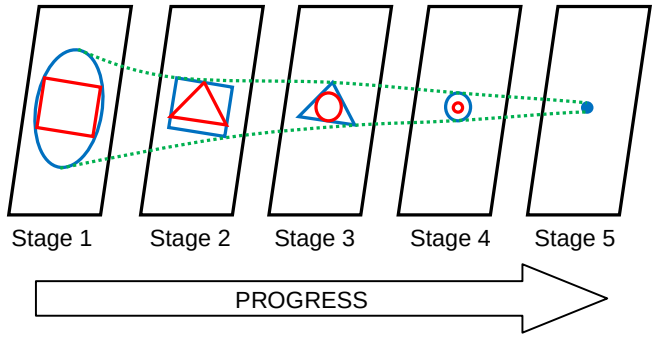

Figure 2. Progressive convergence in SBD

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 [19–22]. CSP is a concept that is often employed in artificial intelligence, operational research and logic programming [23]. 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)* [24]:

$V = \{v_1, \dots, v_n\}$ 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 [24] and Mackworth [25] or continuous variables as shown by Faltings [26]. Dynamic CSP allows adding or removing constraints to the problem model when the problem is not static [27]. 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 [19,28–30].

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 Fig. 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

[28] 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 [31], 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.

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 [32,33]. 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.

3.1 Dynamic Sub-Problem and SBD Process

The design responsibility of a design actor is limited by its sub-problem. Figure 3 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.

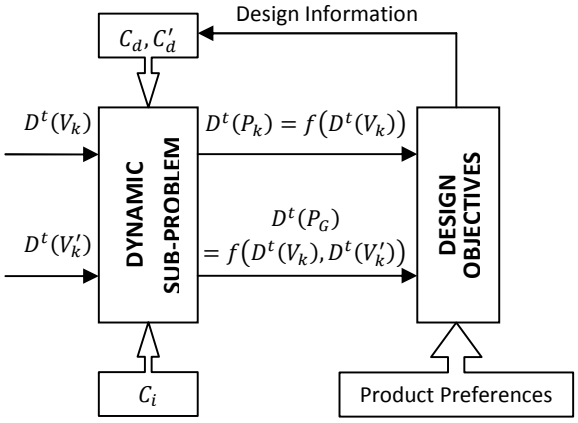

Figure 3. Dynamic sub-problem

The dynamic design process is shown in Fig. 4. 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 Fig. 4. 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.

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.

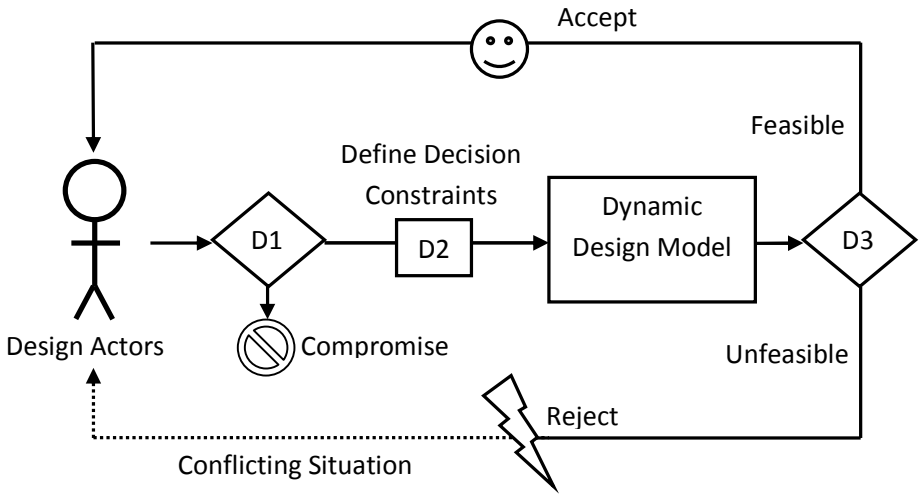

Figure 4. Dynamic SBD process

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 [34]. 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.

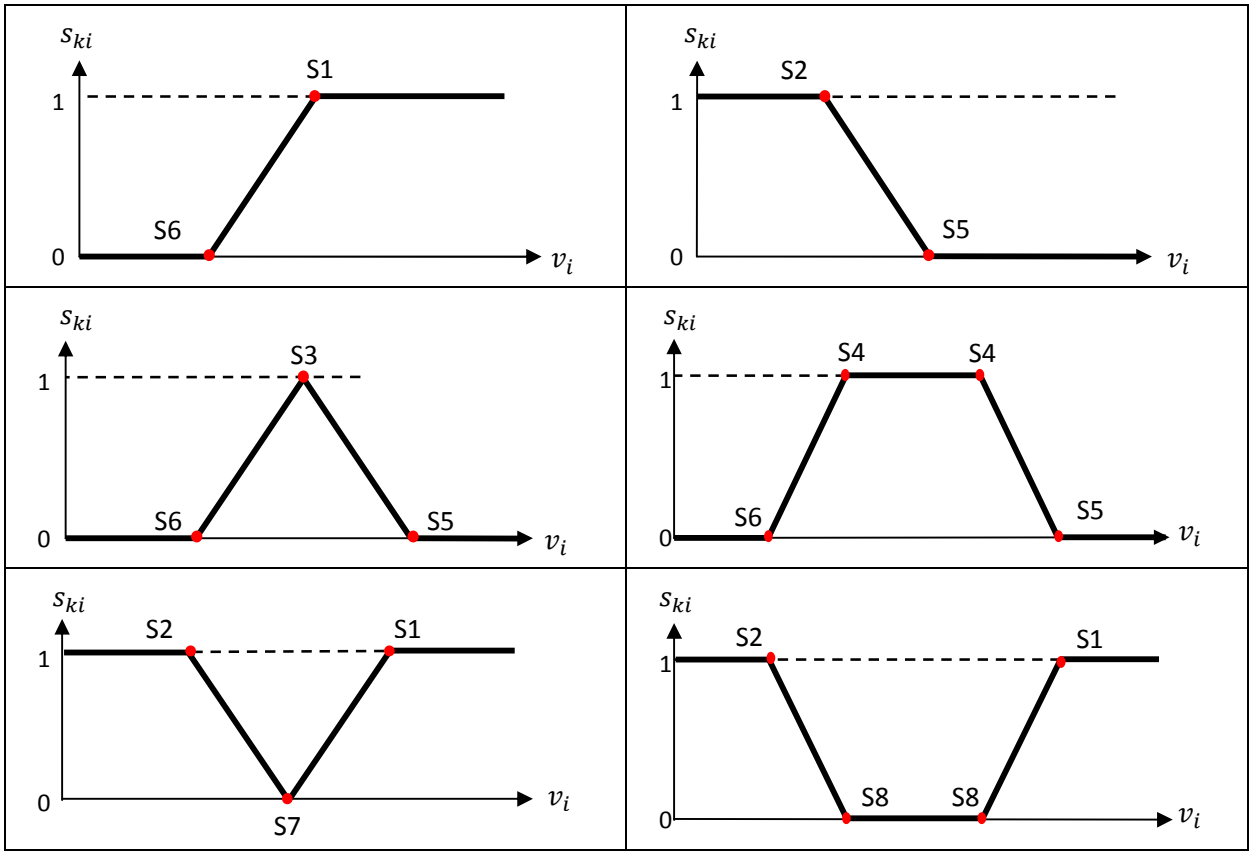

Figure 5. Satisfaction functions

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

If $v_i \geq pref2_{ki}$, $s_{ki} = 0$ (2)

If $pref1_{ki} < v_i < pref2_{ki}$, $1 > s_{ki} > 0$ (Linear) (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 6 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.

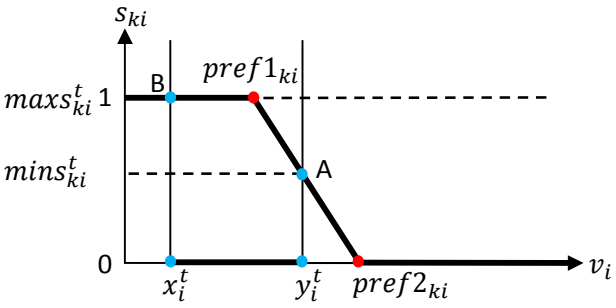

Figure 6. Intervals on the satisfaction function

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. 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:

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

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

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

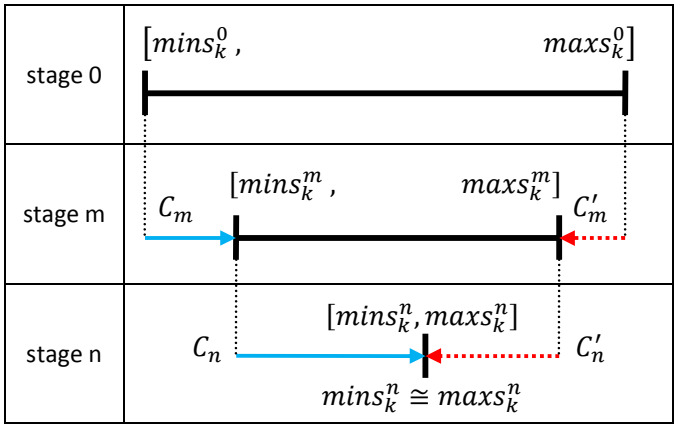

Figure 7. Bilateral convergence of satisfaction interval

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 7 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 Fig. 4.

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. (7), s_k is normalized by T_k , and this provides the wellbeing indicator of actor *k* (wb_k). Figure 8 shows how wb_k is derived by using the example explained with Eqs.

(1-3) 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. (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.

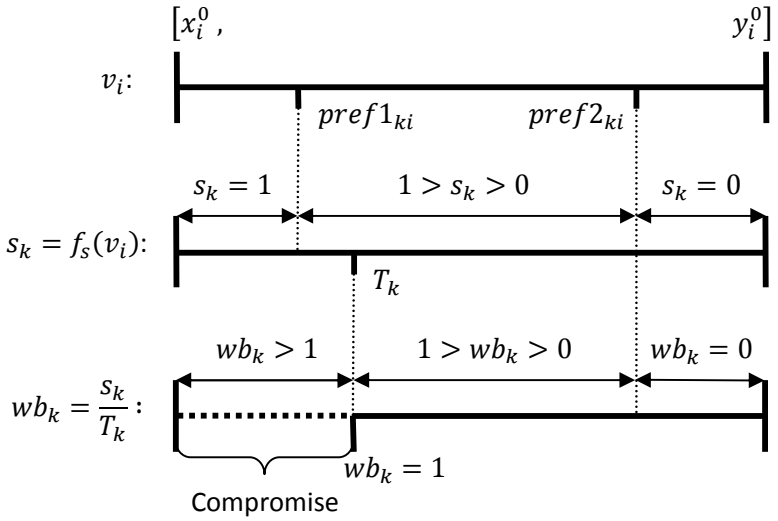

Figure 8. Derivation of wellbeing indicator

$$wb_k = s_k T_k \quad (7)$$

$$wb_k + dv_k^- - dv_k^+ = 1 \quad (8)$$

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

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

4 CP Simulation Process

Canbaz B., Yannou B., Yvars P.-A., (2013), *Improving Process Performance of Distributed Set-based Design Systems by Controlling Wellbeing Indicators of Design Actors.* Journal of Mechanical Design.

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 Fig. 9. 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 [35]. 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 $M_{kj} > 0 \ \ \forall k, i, j$. This coefficient is the design attitude of player *k* that determines how the decision constraint is defined. This is the D3 shown in Fig. 4. 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: $Vn_j \geq minVn_j^t \times \left(1 + M_{kj}\right)$ 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 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.

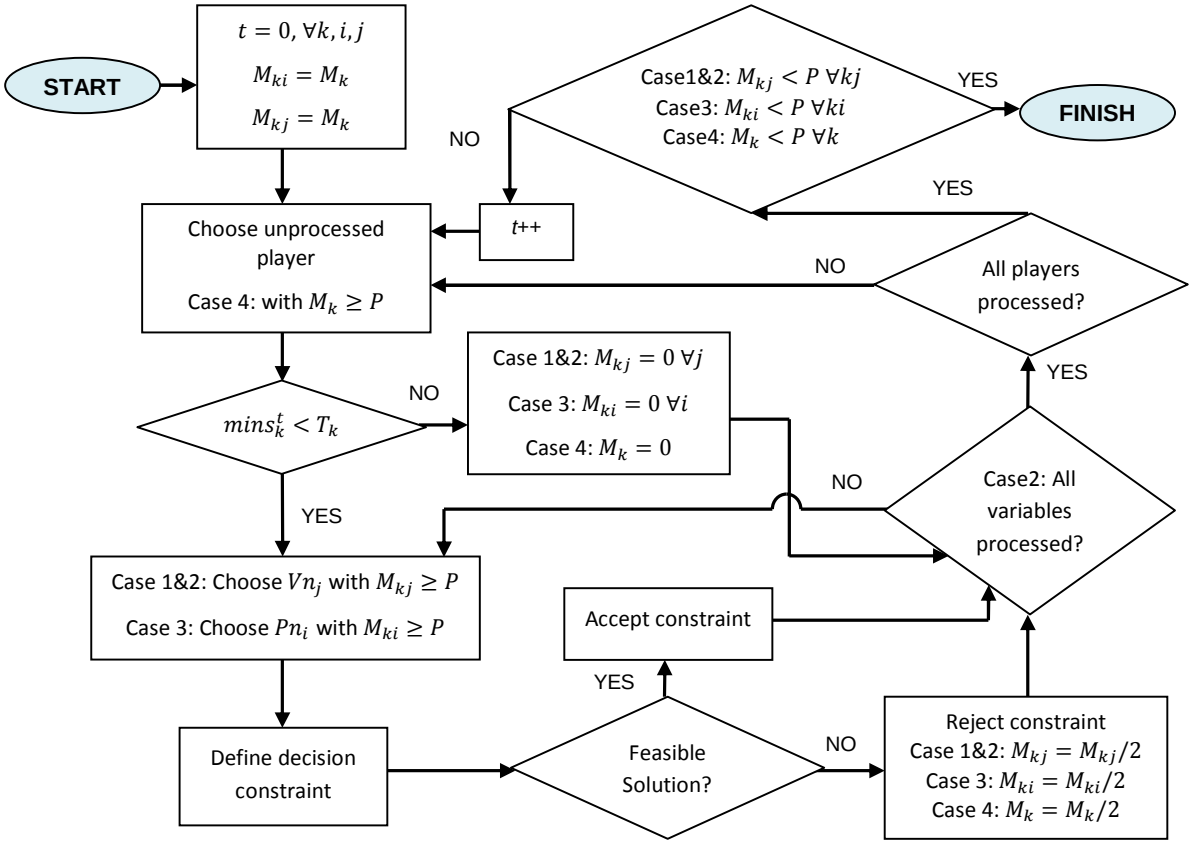

Figure 9. Simulation algorithm

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. (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. (9) represent a vector $D(d_1, \dots, d_n)$. 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. (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.

$$d_i = |s_k - s_{k'}|\,, \forall k, k' \in \mathbb{Z}^+, k' > k \qquad (9)$$

$$Divergence = \sqrt{\sum_{i=1}^{n} (d_i)^2} \qquad (10)$$

5 Monte Carlo Simulations

We performed a Monte Carlo simulation with the design problem of a multi-clutch system derived from the example studied in [36]. Three different player characters are defined with combinations of T_k and M_k design attitudes as shown in Table 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. 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 [37]) 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 Fig. 4 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 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)

5.1 Simulation Problem

The simulation problem is a design problem of a multi-clutch system that connects a weight lifter with an engine, followed by a gearbox (Fig. 10). This is a complex design problem which contains 81 variables and 64 initial constraints. Problem nomenclature and constant values are given in Table 2. There are four designers associated to the problem. Their sub-problems are given in Table 3. The piecewise constraints representing product preferences are shown in Table 4. Transitions are considered linear as in Fig. 5. 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: $Max(s_1 + s_2 + s_3 + s_4)$.

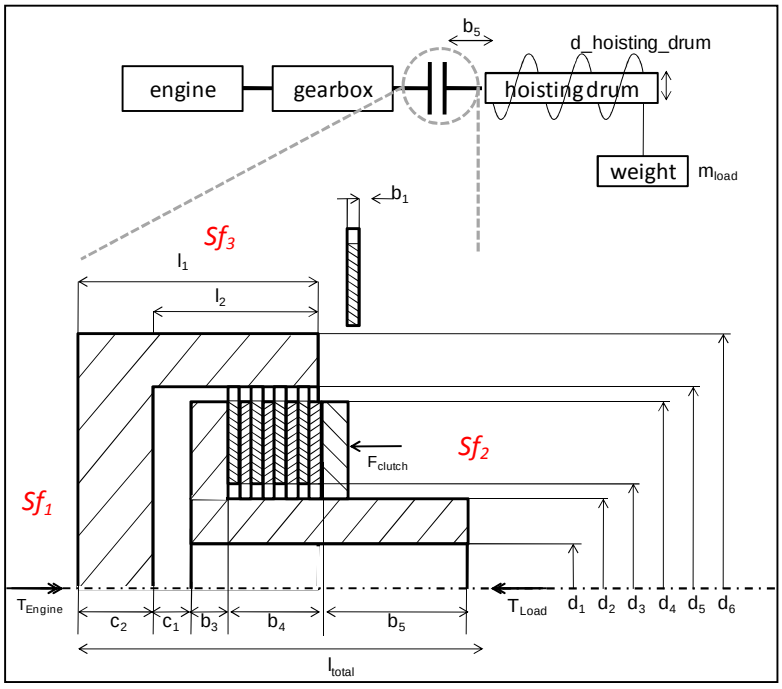

Figure 10. Multi-clutch system

Table 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 3 Clutch sub-problems

	Player1	**Player2**	**Player3**	**Player4**
Performance Variable Objectives	$Max(m_{load})$	$Min(m_s)$	$Min(T)$	$Max(Sf_1, Sf_2, Sf_3, Sf_p)$
Design Variable Objectives	$Max(m_{load})$	$Min(m_e, p_1, p_2, d_2, d_4, d_5, d_6, i, b_1, b_3)$ $Max(d_1, d_3)$	$Min(f, d_m, n, t, i)$ $Max(d_6, b_1, b_3, i)$	$Max(\sigma, P_m)$

Table 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 $2 < Sf_p < 10$, $0 > s_{44} > 1$

5.2 Simulation Results

The average results of 1000 Monte Carlo simulations of each multi-clutch problem case are shown in Fig. 11. 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. 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.

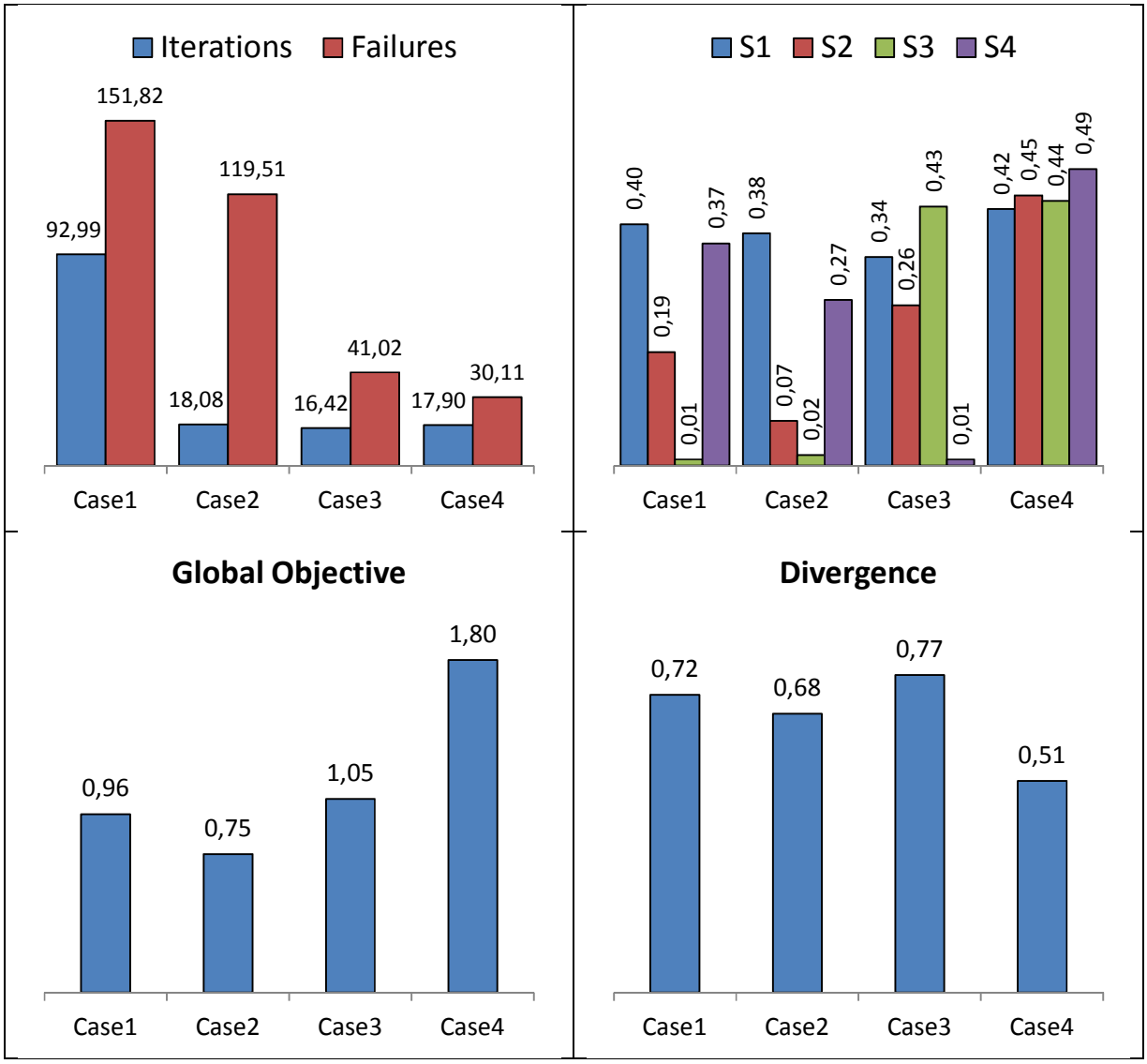

Figure 11. Average results of multi-clutch simulations

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.

Canbaz B., Yannou B., Yvars P.-A., (2013), *Improving Process Performance of Distributed Set-based Design Systems by Controlling Wellbeing Indicators of Design Actors.* Journal of Mechanical Design.

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 future 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.

Canbaz B., Yannou B., Yvars P.-A., (2013), *Improving Process Performance of Distributed Set-based Design Systems by Controlling Wellbeing Indicators of Design Actors.* Journal of Mechanical Design.

References

[1] Papalambros P. Y., Michelena N. F., and Kikuchi N., 1997, "Distributed Cooperative Systems Design," PROCEEDINGS OF THE 11 TH INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, **2**, pp. 265–270.

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

[3] Cramer E., Dennis J., Frank P., Lewis R., and Shubin G., 1994, "Problem Formulation for Multidisciplinary Optimization," SIAM Journal on Optimization, **4**(4), pp. 754–776.

[4] Sobieszczanski-Sobieski J., and Haftka R. T., 1997, "Multidisciplinary aerospace design optimization: survey of recent developments," Structural Optimization, **14**(1), pp. 1–23.

[5] Balling R. J., and Sobieszczanski-Sobieski J., 1996, "Optimization of coupled systems - A critical overview of approaches," AIAA Journal, **34**(1), pp. 6–17.

[6] Lewis K., and Mistree F., 1998, "Collaborative, Sequential, and Isolated Decisions in Design," Journal of Mechanical Design, **120**(4), p. 643.

[7] Zhao L., and Jin Y., 2003, "Work Structure Based Collaborative Engineering Design," ASME Conference Proceedings, (37017b), pp. 865–874.

[8] Chanron V., and Lewis K., 2005, "A study of convergence in decentralized design processes," Res Eng Design, **16**(3), pp. 133–145.

[9] Kim H. M., Michelena N. F., Papalambros P. Y., and Jiang T., 2003, "Target Cascading in Optimal System Design," Journal of Mechanical Design, **125**(3), p. 474.

[10] Park H., Michelena N., Kulkarni D., and Papalambros P. Y., 2001, "Convergence Criteria for Hierarchical Overlapping Coordination of Linearly Constrained Convex Design Problems," Computational Optimization and Applications, **18**(3), pp. 273–293.

[11] Malak R. J., Aughenbaugh J. M., and Paredis C. J. J., 2009, "Multi-attribute utility analysis in set-based conceptual design," Computer-Aided Design, **41**(3), pp. 214–227.

[12] Fathianathan M., and 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**(5), pp. 547–560.

[13] Antonsson E. K., and Otto K. N., 1995, "Imprecision in Engineering Design," Journal of Mechanical Design, **117**(B), pp. 25–32.

[14] Parsons M. G., Singer D. J., and Sauter J. A., 1999, "A HYBRID AGENT APPROACH FOR SET-BASED CONCEPTUAL SHIP DESIGN," Proceedings of 10th International Conference on Computer Applications in Shipbuilding, Cambridge, MA.

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

[16] Sobek D. K., Ward A. C., and Liker J., 1999, "Toyota's Principles of Set-Based Concurrent Engineering," Sloan Management Review, **40**(2), pp. 67–83.

[17] McKenney T. A., Kemink L. F., and Singer D. J., 2011, "Adapting to Changes in Design Requirements Using Set-Based Design," Naval Engineers Journal, **123**(3), pp. 66–77.

[18] Wang J., and Terpenny J., 2003, "Interactive evolutionary solution synthesis in fuzzy set-based preliminary engineering design," Journal of Intelligent Manufacturing, **14**(2), pp. 153–167.

[19] Lottaz C., Smith I. F. ., Robert-Nicoud Y., and Faltings B. ., 2000, "Constraint-based support for negotiation in collaborative design," Artificial Intelligence in Engineering, **14**(3), pp. 261–280.

[20] Panchal J. H., Fernández M. G., Paredis C. J. J., Allen J. K., and Mistree F., 2007, "An Interval-based Constraint Satisfaction (IBCS) Method for Decentralized, Collaborative Multifunctional Design," Concurrent Engineering, **15**(3), pp. 309–323.

Canbaz B., Yannou B., Yvars P.-A., (2013), *Improving Process Performance of Distributed Set-based Design Systems by Controlling Wellbeing Indicators of Design Actors.* Journal of Mechanical Design.

[21] 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**(6), pp. 961–970.

[22] Yannou B., Yvars P.-A., Hoyle C., and Chen W., 2013, "Set-based design by simulation of usage scenario coverage," Journal of Engineering Design, **0**(0), pp. 1–29.

[23] Brailsford S. C., Potts C. N., and Smith B. M., 1999, "Constraint satisfaction problems: Algorithms and applications," European Journal of Operational Research, **119**(3), pp. 557–581.

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

[25] Mackworth A. K., 1977, "Consistency in networks of relations," Artificial Intelligence, **8**(1), pp. 99–118.

[26] Faltings B., 1994, "Arc-consistency for continuous variables," Artificial Intelligence, **65**(2), pp. 363–376.

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

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

[29] Lottaz C S. R. and S. I., 2000, "Increasing Understanding During Collaboration Through Advanced Representations" [Online]. Available: http://www.itcon.org/cgi-bin/works/Show?2000_1. [Accessed: 14-May-2013].

[30] Sam-Haroud D., and Faltings B., 1996, "Consistency techniques for continuous constraints," Constraints, **1**(1-2), pp. 85–118.

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

[32] Canbaz B., Yannou B., and Yvars P.-A., 2011, "A New Framework for Collaborative Set-Based Design: Application to the Design Problem of a Hollow Cylindrical Cantilever Beam," pp. 197–206.

[33] Canbaz B., and Yannou B., 2012, "Constraint Programming Simulation of a Distributed Set- Based Design Framework with Control Indicators," ASME Design Engineering Technical Conferences, Chicago, IL.

[34] Messac A., 1996, "Physical programming - Effective optimization for computational design," AIAA Journal, **34**(1), pp. 149–158.

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

[36] Yannou B., Mazur C., and Yvars P.-A., 2010, "Parameterization and Dimensioning of the Multi-Disc Clutch in the CO4 Environment."

[37] IBM, 2012, "IBM ILOG CPLEX CP Optimizer for Constraint Programs - Features and benefits" [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/about/. [Accessed: 24-Feb-2013].