International Journal of Industrial Engineering, 19(12), 464-474, 2012.

COMPATIBLE COMPONENT SELECTION UNDER UNCERTAINTY VIA EXTENDED CONSTRAINT SATISFACTION APPROACH

Duck Young Kim[1], Paul Xirouchakis[2], Young Jun Son[3]
[1]Ulsan National Institute of Science and Technology, Republic of Korea
[2]École Polytechnique Fédérale de Lausanne, Switzerland
[3]University of Arizona, United States
Corresponding author email: dykim@unist.ac.kr

This paper deals with compatible component selection problems, where the goal is to find combinations of components satisfying design constraints given a product structure, component alternatives available in design catalogue for each subsystem of the product, and a preliminary design constraint. An extended Constraint Satisfaction Problem (CSP) is introduced to solve component selection problems considering uncertainty in the values of design variables. To handle a large number of all possible combinations of components, the paper proposes a systematic filtering procedure and an efficient method to estimate a complex feasible design space to facilitate selection of component combinations having more feasible solutions. The proposed approach is illustrated and demonstrated with a robotic vacuum cleaner design example.

Keywords: Component Selection, Configuration, Design Constraint, Constraint Satisfaction Problem, Filtering

(Received 24 Apr 2012; Accepted in revised form 1 Nov 2012)

1. INTRODUCTION AND BACKGROUND

The product design process involves four main phases: (1) product specification, (2) conceptual design, (3) embodiment design, and (4) detailed design. At each phase, design teams first generate or search for several design alternatives, and select the best one considering design criteria and constraints. In conceptual design, for instance, this generation and selection process consists of four main steps (Pahl and Beitz, 1988) (see Figure 1): (1) decomposition-establish a function structure of a product, (2) definition-search for components to fulfil the sub-functions and define a preliminary design constraint, (3) filtering-combine components to fulfil the overall function, select suitable combinations, and firm up into concept variants, and (4) selection-evaluate concept variants against technical and economic design criteria and select the best one. This divergence and convergence of the search space in design is intended to allow design teams to have unrestrained creativity by producing many initial component alternatives for subsystems, as well as to support the filtering and selection processes to find best design alternatives for a product.

The focus of this paper is on "filterning" in Figure 1. In particular, we consdider a constraint based compatible component selection problem under uncertainty in the values of design variables, especially in redesign and variant design environments. This problem is a combinatorial selection problem, where a component satisfying design constraints is chosen for each subsystem from a pre-defined set (i.e. design catalogue). It is compounded by multiple values or continuous space of design variables and discrete choices of components. In this work, it is assumed that a design catalogue (containing component alternatives for subsystems comprising a product) and a preliminary design constraint are given as input information (see Table 3). By generalizing the problem characteristics found, we formulate the constraint based component selection problem with an extended Constraint Satisfaction Problem (CSP). Finally, a systematic filtering procedure and an efficient method to estimate a complex feasible design space are proposed to select component combinations having more feasible solutions.

A product usually consists of a number of subsystems (see Table 1), where each of them has its own design alternatives, namely component alternatives. Table 2 lists a major list of variables used in this paper. Any combination of components of all subsystems can be a potential design alternative for a product. The selected components must be mutually compatible to achieve the overall functionality, where compatibility means that when components are designed to work others without adjustment (Patel et al., 2003). Therefore, design teams need to find the compatible combinations of components satisfying design constraints from all possible combinations.

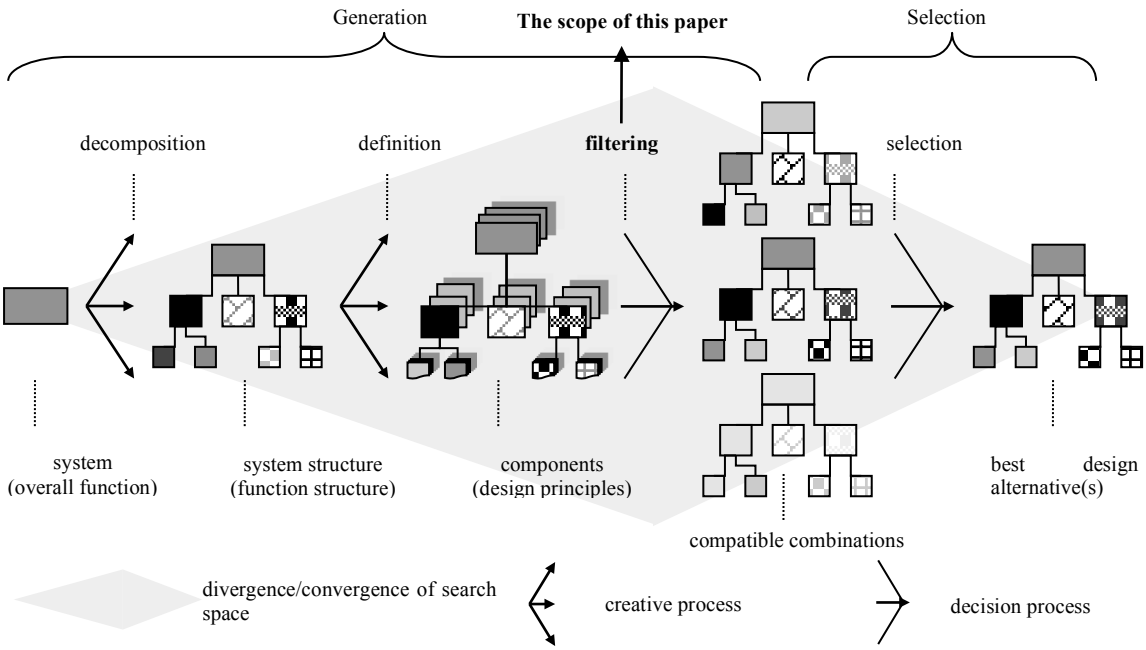

Figure 1. Generation and selection processes in conceptual design (Pahl and Beitz, 1988)

Table 1. A simple selection chart for a compatible combination search problem

Subsystem (F)		Component Alternative (CA)					
		1	*2*	…	*j*	…	*m(i)*
1	F_1	CA_{11}	CA_{12}	…	CA_{1j}	…	$CA_{1m(1)}$
2	F_2	CA_{21}	CA_{22}	…	CA_{2j}	…	$CA_{2m(2)}$
⋮	⋮	⋮	⋮		⋮		⋮
I	F_i	CA_{i1}	CA_{i2}	…	CA_{ij}	…	$CA_{im(i)}$
⋮	⋮	⋮	⋮		⋮		⋮
n	F_n	CA_{n1}	CA_{n2}	…	CA_{ni}	…	$CA_{nm(n)}$

The shading in the cells indicates an exemplary compatible combination of components

Table 2. Nomenclature of variables used in the work

Variable	Description	Variable	Description
F_i	Subsystem i, where i=1, …, n	d_j ($x_{i,k}$)	Domain of $x_{i,k}$ given by component alternative j
$CA_{i,j}$	Component alternative (CA) j for subsystem i, where j=1, …, $m(i)$	$UDO_{i,k}$	United domain across all component alternatives of global design variable k for subsystem i
$x_{i,k}$	Global design variable k for subsystem i	$RDO_{i,k}$	Modified domain of $UDO_{i,k}$ applying AC-3 technique (see Section 3.1)
T ($=\prod_{i=1}^{n} m(i)$)	Number of all possible combination of components	ldo_p	A set of domain, $d_{j(p,i)}(x_{i,k})$ for all i, $j(p,i)$, k in combination(p), where p=1, …, T
j (p, i) ∈ {1, … $m(i)$}	Considered CA for subsystem i in combination p, where p=1, …, T	CS (comb.(p))	Compatibility score for combination p

However, the number of possible combinations grows rapidly with both the number of subsystems as well as the number of component alternatives for each subsystem. Furthermore, components are closely coupled with each other, therefore a change in one component may affect performance of the other components in the combination (Carlson-Skalak, 1996). Singhal and Singhal (1996) showed that the number of feasible designs decreases exponentially with the increase in the number of incompatible pairs of components. For these reasons, the first few compatible combinations found are usually accepted without further evaluation in many component selection design studies (Bradley and Agogino, 1994; Darr and Birmingham, 2000). However, this strategy may (1) omit more valuable combinations and (2) cause difficulties for design teams in selecting the best design alternative of a product satisfying multiple design criteria from a small set of compatible combinations found.

Researchers in the field of Artificial Intelligence have focused mainly on the dynamic nature of configuration with a relatively simplified binary design constraints, based on the notion of 'Conditional CSP' (Mittal and Falkenhainer, 1990; Sabin and Freuder, 1996). Kim et al. (2006) pointed out that this type of design constraints imply that the compatibility of a certain pair of components is given a priori, and each component itself can be considered as a design variable. In other words, no matter what and how many complex binary design constraints are involved in a certain pair of components, they are simplified into a 'good or 'bad' relation of the pair, without directly considering coupled design variables of components and the constraints behind them. This paper therefore introduces an extend CSP to measure the compatibility of components and consequently filter out compatible combinations of components efficiently.

2. COMPATIBLE COMPONENT SELECTION PROBLEM UNDER UNCERTAINTY

2.1 Real-life Design Example, Design Catalogue, and Assumptions

This section describes a robotic vacuum cleaner (RVAC) design example (see Figure 2), which contains four subsystems (see Table 3): F_1: Filter, F_2: Main motor & Impeller, F_3: Battery, and F_4: Dust container & Cover. Each subsystem has two component alternatives. For example, 'Regular' and 'HEPA' are alternatives for the subsystem 'Filter'. As a result, there are 16 different combinations of components, i.e. 16 design alternatives for a RVAC. For example, the combination of HEPA, Type A, Li-ion, and Cyclone constitues a design alternative. A main task of compatible component selection addressed in this paper is to find the compatible combinations considering the design constraints. Note that the desired number of combinations can be specified a priori, and the selected combinations will be used as input for the final selection process (see Figure 1).

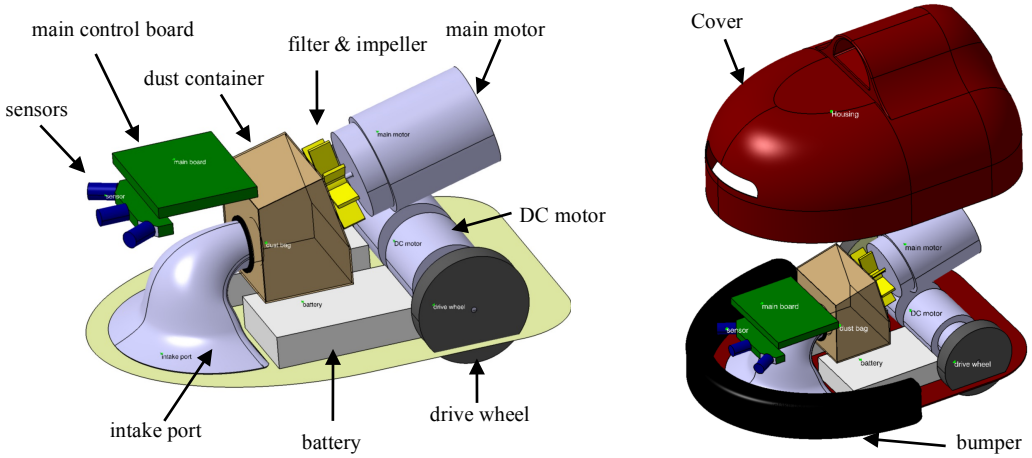

Figure 2. Case example: a robotic vacuum cleaner (RVAC)

It is important to note that each component alternative has different values for the global design variables (synonymous with interface variables, shared variables, or dependent variables in the literature). For example, the airflow resistance of 'HEPA' filters is usually bigger than that of 'Regular' filters. Furthermore, each global design

variable has its own domain because each design team may have a set of approximate, admissible and achievable values for a global design variable, e.g. {3,4,5,6,7} of the airflow resistance of 'Regular' filter in Table 3. It can be said, especially in the early design phase, that various factors such as lack of knowledge, incomplete information, and open-ended creativity entail uncertainty in the values of design variables. In this paper, it is assumed that the domain of each global design variable is either a set of discrete choices or bound within an interval. In the RVAC example, a further assumption is made that an interval is discretized and approximated as a finite set of discrete choices. For example, the original interval, [3…7] is discretized into {3,4,5,6,7} for the domain of the airflow resistance of 'Regular' filter. In engineering problems, there is usually a maximal precision such as nanometer and microgram, and design variable values can be discretized into the maximal useful resolution (Sam-Haroud and Faltings, 1996).

In addition, it is assumed that in the design constraint C4 in Table 3, the values for the energy consumption (Watt·h) of F_2 are approximated by Watt(N·m/s)=airflow(m^3/s) × suction pressure(N/m^2); Time(h)=Ah × V/Watt; neglecting losses we can express motor power=airflow × suction pressure; assuming an airflow of 0.1 m^3/sec, then constraint C4 expresses our desire for assuring an operating time of at least 0.5h, Capacity (Ah·V) ≥ motor power×0.5h=(0.1× $\boldsymbol{x}_{2,1}$) × 0.5h.

Design variables and constraints are continuously changed by insert, delete, and modify operations throughout the design process because of the evolutionary and creative nature of the design process. Nevertheless, once a valid system structure is established such that a unique fixed configuration is specified, it is possible to assume that the design constraints are static in contrast with the dynamic constraints in Mittal and Falkenhainer (1990). In other words, a component selection problem is not necessarily considered as a Conditional CSP of Mittal and Falkenhainer (1990). Moreover, in a collaborative design environment, different design teams participate and collaborate all together from the beginning of the design process, and thus, each subsystem is developed in parallel by design teams in such a way that the properties of components are more or less well-known and quantitatively elaborated. This is even true in redesign or variant design. Consequently, a rough set of design variables and their relationships can be specified to some degree, and this rough information (preliminary design constraint) is still valuable to eliminate incompatible components from the search space.

Table 3. Design catalogue for a robotic vacuum cleaner

Subsystem F_i	Global design variable $x_{i,k}$	Component alternatives $CA_{i,j}$ and Domain $d_j(x_{i,k})$		Unified Domain $UDO_{i,k}$
F_1: Filter		**$CA_{1,1}$: Regular**	**$CA_{1,2}$: HEPA**	
	$\boldsymbol{x}_{1,1}$: Airflow resistance(%)	$d_1(x_{1,1})$:{3,4,…,7}	$d_2(x_{1,1})$:{5,6,…,12}	$UDO_{1,1}$:{3,4,…,12}
	$\boldsymbol{x}_{1,2}$: EOL value(€)	$d_1(x_{1,2})$:{5,6,…,15}	$d_2(x_{1,2})$:{15,16,…,25}	$UDO_{1,2}$:{5,6,…,25}
F_2: Main motor & Impeller		**$CA_{2,1}$: Type A**	**$CA_{2,2}$: Type B**	
	$\boldsymbol{x}_{2,1}$: Suction (Pa)	{600,601,…,900}	{850,851,…,1200}	{600,601,…,1200}
	$\boldsymbol{x}_{2,2}$: layout type (motor axis-airflow direction)	{H-H,V-V}	{V-H}	{H-H, V-H, V-V}
F_3: Battery		**$CA_{3,1}$: Ni-Cd**	**$CA_{3,2}$: Li-ion**	
	$\boldsymbol{x}_{3,1}$: Capacity(Ah×V)	{15,16,…,30}	{20,21,…,50}	{15,16,…,50}
F_4: Dust container & Cover		**$CA_{4,1}$: Dust bag or plastic bin**	**$CA_{4,2}$: Cyclone**	
	$\boldsymbol{x}_{4,1}$: Housing type	{flat,semi}	{top}	{flat,semi,top}
	$\boldsymbol{x}_{4,2}$: EOL value(€)	{5,10,15,20}	{15,20,25}	{5,10,15,20,25}
Design constraints	C1: $\boldsymbol{x}_{2,1}$ × (100%- $\boldsymbol{x}_{1,1}$) ≥ 700Pa (suction) C2: $\boldsymbol{x}_{1,2}$ + $\boldsymbol{x}_{4,2}$ ≥ 40€ (end of lifecycle value) C3: if $\boldsymbol{x}_{2,2}$ = V-V or V-H then $\boldsymbol{x}_{4,1}$ = top (topological) C4: $\boldsymbol{x}_{3,1}$ ≥ 0.5×0.1× $\boldsymbol{x}_{2,1}$ (Operating time ≥ 0.5hr.)			

(Airflow resistance (%): % of suction pressure)

2.2 Design Space

A special property of component selection problems is that each combination of components creates its own design constraint having a different set of domains for the global design variables. For example, the domains of $\boldsymbol{x_{1,1}}$: Airflow resistance and $\boldsymbol{x_{1,2}}$: EOL value composed by the following components: $CA_{1,1}$ (Regular), $CA_{2,1}$, $CA_{3,1}$, and $CA_{4,1}$ are {3,4,5,6,7} and {5,10,15}, respectively, while they are {5,6,7,…,12} and {15,20,25} in the case composed by $CA_{1,2}$(HEPA), $CA_{2,1}$, $CA_{3,1}$, and $CA_{4,1}$ in Table 3. In other words, for the RVAC design example, we need to deal with 16 different cases (i.e. CSPs; see Section 2.3) each having different domains but the same set of design constraints. Note that a combination p of components is denoted as combination(p) such that

$$\text{combination}(p) = \text{CA}_{1,j(p,1)} \times \text{CA}_{2,j(p,2)} \times \text{K CA}_{i,j(p,i)} \text{K} \times \text{CA}_{n,j(p,n)}$$

For example, $j(p,1) = 2, j(p,2) = 1, j(p,3) = 1, j(p,4) = 2$ in combination(p) = $CA_{1,2}$×$CA_{2,1}$× $CA_{3,1}$×$CA_{4,2}$.

In the last column of Table 3, $UDO_{i,k}$ implies all possible values of each global design variable across all components, and this can be obtained by the union of domains: $UDO_{i,k} = \bigcup_{j=1}^{m(i)} d_j(x_{i,k})$. For example, $UDO_{1,1}$ = $d_1(x_{1,1}) \cup d_2(x_{1,1})$ = {3,4,5,6,7} ∪ {5,6,7,…,12} = {3,4,5,…,12}.

The global design space of a compatible component selection problem is defined by the Cartesian product of $UDO_{i,k}$ for all subsystem i and global design variable k. Note that hereafter **UDO** denotes a set of unified domains, $UDO_{i,k}$ for all i and k. For example, **UDO** of the RVAC design example is {{3,4,5,…,12}, {5,6,7…,25}, {600,601,602,…,1200}, {H-H, V-H, V-V}, {15,16,17,…,50}, {flat,semi,top}{5,10,15,20,25}}. Similarly, the local design space of a combination(p) is defined by the Cartesian product of $d_{j(p,i)}(x_{i,k})$ for all i and k, and the selected component $j(p,i)$ for each subsystem. Note that hereafter $\boldsymbol{ldo}_p$ denotes a set of domains, $d_{j(p,i)}(x_{i,k})$ for all i, $j(p,i)$, k in combination(p). For example, for combination(1) = $CA_{1,1}$×$CA_{2,1}$×$CA_{3,1}$×$CA_{4,1}$, $\boldsymbol{ldo}_1$ = {{3,4,5,6,7}, {5,6,7…,15}, {600,601,602,…, 900}, {H-H, V-V}, {15,16,17,…,30}, {flat,semi}{5,10,15,20}}. And, **LDO** contains sets of all $\boldsymbol{ldo}_p$, i.e. {$\boldsymbol{ldo}_1, \boldsymbol{ldo}_2, \ldots, \boldsymbol{ldo}_p, \ldots, \boldsymbol{ldo}_T$}.

2.3 CSP Formation for Compatible Component Selection

A Constraint Satisfaction Problem (CSP) is formally defined as follows (Tsang, 1993):

Definition 1. A CSP involves a triple <**X**, **D**, **C**>, where **X** denotes a set of finite variables, **D**: denotes a set of domains containing a finite set of values for each variable, and **C**: denotes a finite set of constraints that restrict the combination of values from the Cartesian product of all variable domains. In this work, a CSP consisting of **X**, **D**, and **C** is denoted as CSP<**X**,**D**,**C**>.

Three general tasks of CSPs are: (1) to find one tuple of $\boldsymbol{D}$, satisfying $\boldsymbol{C}$, (2) to find all tuples of $\boldsymbol{D}$, satisfying $\boldsymbol{C}$, or (3) to validate the feasibility of a certain tuple. A tuple is a set of values assigned to each variable from the corresponding domain. Therefore, each tuple is a candidate solution for a CSP, and those tuples that satisfies all the constraints are the solutions. Among the three general tasks, the second one is more relevant for the context of compatible component selection. For the first and third cases, a corresponding task in component selection is to find combinations of components having a feasible tuple, where values are already assigned for all design variables, and therefore there is no need for further design (e.g. no room for optimisation). In addition, if there are many combinations each having a feasible tuple, it is difficult to judge which one is the best. Furthermore, many preliminary design problems involving incomplete data and partially defined variables are usually under-constrained (Medland and Mullineux, 2000), therefore many combinations could have many feasible tuples. Therefore, a specific goal in this paper is to find the combination that has more feasible tuples than the others as illustrated in Figure 3. The reasoning behind this goal is that if the combination found has more feasible tuples, there will be more chance to control the design variables for optimisation based on design criteria in the subsequent design stages. Given that $\boldsymbol{X}$ and $\boldsymbol{C}$ denote the sets of global design variables and design constraints, respectively, compatible component selection problems are formally defined as follows:

Definition 2. A compatible component selection problem is a quadruple <**X, LDO, UDO, C**>, where **LDO** = {$\mathbf{ldo}_1$, $\mathbf{ldo}_2$, …, $\mathbf{ldo}_p$, …, $\mathbf{ldo}_T$} is a set of T subsets of **UDO,** such that CSP<**X**, $\mathbf{ldo}_p$, **C**> ⊆ CSP<**X**, **UDO**, **C**>, ∀p=1,2,…,T.

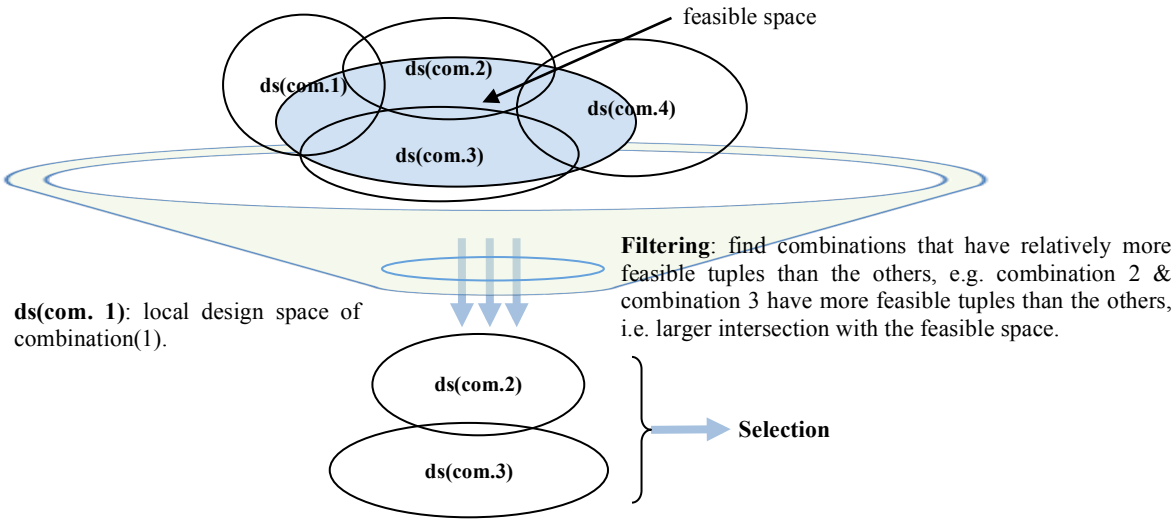

Figure 3. The filtering process (compatible component selection; see Figure 1) in the CSP structure

2.4 Computational Complexity

As mentioned in Section 2.2, the RVAC design example involves 16 possible combinations (i.e. CSPs). To emplay an exact solution approach, it is necessary to find all feasible tuples for each combination exhaustively (NP-hard in general) and rank the combinations based on the number of feasible tuples in each combination. In other words, a compatible component selection problem consists of 'T' NP-hard problems (i.e. the number of total combinations T increases exponentially with the numbers of subsystems and component alternatives of subsystems as shown in Table 1) and one sorting problem. A possible brute force solution method is to employ techniques estimating feasible number of solutions (Dechter and Pearl, 1988; Bailleux and Chabrier, 1997) to all CSPs. However, Roth (1996) discussed the computational complexity of counting problems and proved that the problem of computing the number of solutions of a (binary) CSP is complete for #P, where #P problems ask 'how many' rather than 'are there any', i.e. counting problems to find or approximate the number of solutions in the decision problems in the set NP (e.g. CSP). Clearly, a #P problem must be at least as hard as the corresponding NP problem (http://en.wikipedia.org/wiki/Sharp-P). Furthermore, approximating this number within a reasonable error bound is NP-hard. Therefore, the above techniques are generally intractable due to too many CSPs available in a compatible component selection problem.

3. PROPOSED FILTERING PROCESS

This section presents a generic framework for the filtering process as a solution estimation method for the extended CSP formulated for component selection. For the case of (1) relatively smaller number of total combinations or (2) insufficient common values between domains of different components, a possible method is to employ a consistency technique (Mackworth, 1977) for all CSPs, obtain the upper bound on the number of feasible tuples of each CSP, and use these upper bounds to rank order the combinations. In general, however, considering that all different CSPs have the same set of design constraints, many common values among the components (e.g. 5, 6, 7 (airflow resistance) are common between Regular and HEPA filters in Table 3) may cause the combinations to have many redundant feasible tuples, where it would be wasteful to assess each and every combination. In addition, instead of absolute counting or estimating feasible tuples for all combinations, relative comparison of the search space are more efficient to identify the best combination. In other words, since CSP<$\boldsymbol{X}$, $\boldsymbol{ldo}_p$, $\boldsymbol{C}$> $\subseteq$ CSP<$\boldsymbol{X}$, **UDO**, $\boldsymbol{C}$>, the feasible space of CSP<$\boldsymbol{X}$, **UDO**, $\boldsymbol{C}$> will be computed instead of computing the feasible space of each CSP<$\boldsymbol{X}$, ldo_p, $\boldsymbol{C}$> for all p, and relative comparisons between each local design space and the feasible space of CSP<$\boldsymbol{X}$, **UDO**, $\boldsymbol{C}$> will be made.

3.1 Estimation of the Feasible Space of CSP <*X*, UDO, *C*>

Finding an exact feasible space is an extremely hard problem in general. During the last two decades, there have been research efforts to estimate a feasible space depending on the properties of a problem such as continuous/discrete, binary/non-binary, linear/non-linear, and convex/non-convex. The main goal here is to find the most compact bounds of the feasible space (e.g. convex hull of feasible solutions), which conservatively enclose all the solutions (Sam-Haroud and Faltings, 1996).

Yao and Johnson (1997) presented a domain propagation algorithm to obtain a multidimensional compact interval box, by which one can expect a better estimation of the feasible space (see Figure 4-(b)) than by the simple unary projection of constraints (see Figure 4-(a)). Their basic idea is to apply a mathematical programming method to determining the minimum and maximum feasible values of each variable. Under strict assumptions such as the convexity of the search space and nonlinear inequality constraints, Director and Hachtel (1977) proposed a simple polyhedral approximation method based on mathematical programming and Monte Carlo simulation techniques. Interval arithmetic techniques, by their nature, can be applicable to estimating the feasible space under strict assumptions such as polynomial constraints, a matrix made by the constraint coefficients are regular, and globally dominant Jacobians (Neumaier, 1990). For the above specific constraints, interval propagation can be achieved efficiently because it handles the entire set of constraints as a whole and in most cases, only minimum and maximum values are enough to consider for validation (Karni and Belikoff, 1996). Qureshi et al. (2010) integrated a set-based design approach into a statistical robust design method of mechanical systems in order to effectively explore the solution space of the embodiment design phase. Several consistency techniques have been proposed for continuous variables (Lhomme, 1993; Faltings, 1994; Sam-Haroud and Faltings, 1996; Benhamou et al., 1999; Lottaz et al., 2000; Granvilliers and Benhamou, 2006; Bolloju et al., 2012) (see Figure 4-(c)). In fact, consistency techniques (Mackworth, 1977) are usually used in pre-processing to discard many inconsistent values in domains based on the construction of partial solutions in the discrete search space. For the continuous case, interval based algorithms are incorporated into consistency techniques (e.g. Granvilliers and Benhamou, 2006) or continuous variables are discretized and approximated as a finite set of discrete choices (values) (Sam-Haroud and Faltings, 1996).

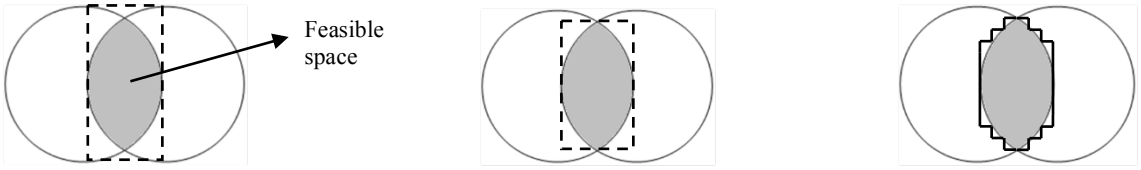

Figure 4. Feasible space approximation techniques

For the case of the RVAC design example, domains of design variables are discretized and the constraints are binary. Therefore, conventional arc-consistency techniques (e.g. AC-3, Mackworth, 1977) can be used to eliminate inconsistent values from the domains to estimate a conservative pave of the feasible space. Depending on the problem size and complexity, stronger consistency techniques (e.g. path-consistency (Han and Lee, 1988)) or weaker consistency techniques (e.g. partial arc-consistency (Nadel, 1989)) can be employed.

Table 4 depicts the revised domains of each $UDO_{i,k}$, denoted as $RDO_{i,k}$, by applying AC-3 which allows all variables and associated constraints to be revised in any order. Then, the feasible space of the RVAC design example can be approximated by the Cartesian product of $RDO_{i,k}$ for all i and k. The upper bound of the number of feasible tuples can be approximated by $\prod_{\forall i,k} |RDO_{i,k}|$, i.e. 10×11×279×3×14×3×3=11,600,820, which is about 5.7% of that of all possible tuples, $\prod_{\forall i,k} |UDO_{i,k}|$ = 204,460,200. Note that the number of elements of a set S is denoted | S |.

Table 4. Revised domain $RDO_{i,k}$ of the RVAC design example after applying AC-3 technique

F_i		$UDO_{i,k}$	$\lvert UDO_{i,k} \rvert$		$RDO_{i,k}$	$\lvert RDO_{i,k} \rvert$
F_1	$UDO_{1,1}$	{3,4,…,12}	10	$RDO_{1,1}$	{3,4,…,12}	10
	$UDO_{1,2}$	{5,6,…,25}	21	$RDO_{1,2}$	{15,16…,25}	11
F_2	$UDO_{2,1}$	{600,601,…,1200}	601	$RDO_{2,1}$	{722,723,…,1000}	279
	$UDO_{2,2}$	{H-H, V-H, V-V}	3	$RDO_{2,2}$	{H-H, V-H, V-V}	3
F_3	$UDO_{3,1}$	{15,16,…,50}	36	$RDO_{3,1}$	{37,38,…,50}	14
F_4	$UDO_{4,1}$	{flat,semi,top}	3	$RDO_{4,1}$	{flat,semi,top}	3
	$UDO_{4,2}$	{5,10,15,20,25}	5	$RDO_{4,2}$	{15,20,25}	3

3.2 Rank Ordering

To find the combination containing the maximum number of feasible tuples, it is necessary to compute the feasible space of CSP<***X***, ldo_p, ***C***> for all p. However, considering a large number of total combinations, relative comparisons are made among each local design space of a combination and the estimated feasible space of CSP<***X***, **UDO**, ***C***> that was obtained in Section 3.1. As illustrated in Figure 3, if the local design space of a combination has the largest intersection with the feasible space of CSP<***X***, **UDO**, ***C***>, the feasible space of that combination is the largest. However, in general, finding this intersection is not straightforward as each local design space restricts a certain region of the feasible space again. Therefore, we propose to estimate the upper bound of the intersection size (i.e. upper bound of the number of feasible tuples of a combination) based on the common values of each design variable between a local design space and the estimated feasible space. In other words, the intersection size, namely, the compatibility score of the combination p, denoted as CS(combination(p)) in this work, can be estimated by the product of the number of common values between $d_{j(p,i)}(x_{i,k})$ and $RDO_{i,k}$ for all i and k as follows:

$$CS(combination(p)) = \prod_{\forall i,k} \left| d_{j(p,i)}(x_{i,k}) \cap RDO_{i,k} \right| \quad \ldots \quad (1)$$

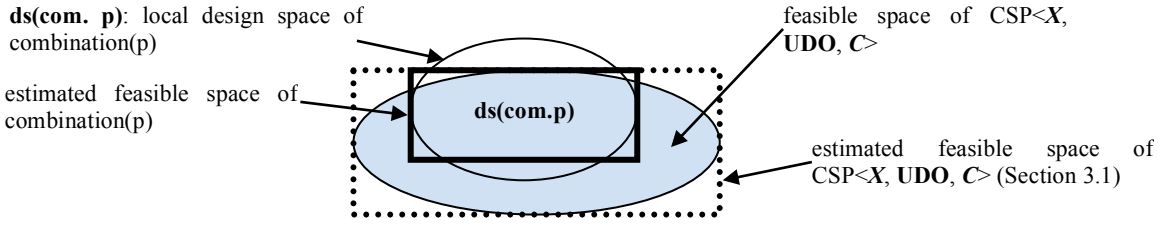

Figure 5. Estimation of the numberof feasible tuples of a combination

As illustrated in Figure 5, the compatibility score means the upper bound of the number of feasible tuples of a combination, which can be obtained by determining the intersection between the local design space of combination(p) and the estimated feasible space of CSP<***X***, **UDO**, ***C***>. For example, CS(*combination(3)* = $CA_{1,1} \times CA_{2,1} \times CA_{3,2} \times CA_{4,1}$) can be calculated by: |{3,4,…,7}| × |{15}| × |{722,723,…,900}| × |{H-H,V-V}| × |{37,38,…,50}| × |{flat,semi}| × |{15,20}| = 5×1×179×2×14×2×2 = 110,240. Accordingly, it can be said that the maximum number of feasible tuples of combination(3) is 110,240.

Table 5. Compatibility scores of all 16 combinations of components

CS(*combination*(*p*))	Calculation		Rank
CS(*combination(1)*=$CA_{1,1}\times CA_{2,1}\times CA_{3,1}\times CA_{4,1}$)	5×11×179×2×0×2×2 =	0	incompatible
CS(*combination(2)*=$CA_{1,1}\times CA_{2,1}\times CA_{3,1}\times CA_{4,2}$)	5×1×179×2×0×1×3 =	0	incompatible
CS(*combination(3)*=$CA_{1,1}\times CA_{2,1}\times CA_{3,2}\times CA_{4,1}$)	5×1×179×2×14×2×2 =	110,240	5
CS(*combination(4)*=$CA_{1,1}\times CA_{2,1}\times CA_{3,2}\times CA_{4,2}$)	5×1×179×2×14×1×3 =	75,180	6
CS(*combination(5)*=$CA_{1,1}\times CA_{2,2}\times CA_{3,1}\times CA_{4,1}$)	5×1×151×1×0×2×2 =	0	incompatible
CS(*combination(6)*=$CA_{1,1}\times CA_{2,2}\times CA_{3,1}\times CA_{4,2}$)	5×1×151×1×0×1×3 =	0	incompatible
CS(*combination(7)*=$CA_{1,1}\times CA_{2,2}\times CA_{3,2}\times CA_{4,1}$)	5×1×151×1×14×2×2 =	42,280	7
CS(*combination(8)*=$CA_{1,1}\times CA_{2,2}\times CA_{3,2}\times CA_{4,2}$)	5×1/×151×1×14×1×3 =	31,710	8
CS(*combination(9)*=$CA_{1,2}\times CA_{2,1}\times CA_{3,1}\times CA_{4,1}$)	8×11×179×2×0×2×2 =	0	incompatible
CS(*combination(10)*=$CA_{1,2}\times CA_{2,1}\times CA_{3,1}\times CA_{4,2}$)	8×11×179×2×0×1×3 =	0	incompatible
CS(*combination(11)*=$CA_{1,2}\times CA_{2,1}\times CA_{3,2}\times CA_{4,1}$)	8×11×179×2×14×2×2 =	1,764,224	1
CS(*combination(12)*=$CA_{1,2}\times CA_{2,1}\times CA_{3,2}\times CA_{4,2}$)	8×11×179×2×14×1×3=	1,323,168	2
CS(*combination(13)*=$CA_{1,2}\times CA_{2,2}\times CA_{3,1}\times CA_{4,1}$)	8×11×151×1×0×2×2 =	0	incompatible
CS(*combination(14)*=$CA_{1,2}\times CA_{2,2}\times CA_{3,1}\times CA_{4,2}$)	8×11×151×1×0×1×3 =	0	incompatible
CS(*combination(15)*=$CA_{1,2}\times CA_{2,2}\times CA_{3,2}\times CA_{4,1}$)	8×11×151×1×14×2×2 =	744,128	3
CS(*combination(16)*=$CA_{1,2}\times CA_{2,2}\times CA_{3,2}\times CA_{4,2}$)	8×11×151×1×14×1×3=	558,096	4

Table 5 summarizes the compatibility scores and their calculation processes. According to the compatibility scores in Table 5, combination(11), the combination of HEPA, Type A, Li-ion, and Dust bag or plastic bin is the best compatible combination, while 8 combinations are incompatible. These 8 combinations having no feasible tuples must be eliminated from further consideration. Note that the desirable number of acceptable combinations can be specified a priori. For example, if the design team specifies this number as '4', combination(11), combination(12), combination(15), and combination(16) will be selected for the subsequent selection process. If there exist a tie in compatibility scores by multiple combinations, the design team can break the tie in an arbitrary manner.

4. DESIGN IMPLICATIONS OF WORK AND DISCUSSION ON PROBLEM VARIATIONS

The goal of this section to discuss validation of the underlying assumptions of the considered problem in this work. In general, each component can have different global design variables in a subsystem, and hence associated design constraints can vary. For example, it may be necessary to consider another design variable, 'Airflow resistance' for 'Dust bag or plastic bin', but this design variable is not required for 'Cyclone' (see Table 3). However, Definition 2 in Section 2.3 already includes this type of complicated cases, where all necessary design variables across different components in a subsystem can be considered as a whole, while the unnecessary design variable 'Airflow resistance' for 'Cyclone' has a NULL value.

Darr and Birmingham (2000) have suggested that if it is possible to eliminate incompatible components or incompatible pairs of components after a few consistency checks, then the possible combinations will be reduced drastically. Hence, it is possible to take further assumptions into account. For example, if global design variables are clearly specified and each design variable has a deterministic value, it is then a conventional component selection problem under certainty (e.g. in Patel et al., 2003).

In general, many real-world design problems involve design team's preferences among potential design solutions,

therefore not all solutions are equally good. For example, a suction motor design team may want to increase the suction power requiring a sufficient battery capacity, which naturally requires a larger size housing design while a housing design team may prefer a smaller size housing design (Kim et al., 2006). Nonetheless, in classical CSPs, there is no notion of 'better' or 'worse' solutions, which is still true for the filtering process. Recently, there have been some research efforts devoted to address this issue of incorporating user preferences into classical CSPs under the following research terms: constraint optimization (Tsang, 1993), soft-constraints (Freuder et al., 2003) and preference based search (Junker, 2002). Patel et al. (2003) have considered the preference on each component and attempted to use physical programming and graph theoretic techniques to obtain sets of preferred subsystem design concepts. D'Ambrosio and Birmingham (1995) presented a preference-directed design method, by which they tried to minimize the enumeration and evaluation of design alternatives.

It can be considered that after defining a feasible space (see Section 3.1), a stakeholder (e.g. a project manager) can distribute this information as a design guideline (e.g. admissible value ranges for design variables) to each subsystem design team. However, in collaborative design, this type of centralised and hierarchical approach would not be recommendable as it can restrain the creativity of each design team. Furthermore, it is very often more difficult to find the feasible space of a problem with universal domains; for the RVAC design example, **UDO** specifies the necessary domains for estimating the feasible space.

5. CONCLUSION

This paper introduced a CSP extension for constraint based compatible component selection problems considering uncertainty in the values of design variables that is modelled by either interval or discrete choices. The considered compatible component selection problem is a combinatorial selection problem to choose a component from a pre-defined set of components for each subsystem of a product satisfying design constraints. The main feature of the problem is that each combination of components creates its own CSP having a different set of domains for the global design variables, and thus the problem includes many CSPs based on the number of possible combinations of components. Therefore, the decision space of the considered problem is compounded by multiple values or continuous space of global design variables and discrete choices of components.

To resolve such challenging component selection problem, a system framework was proposed for the filtering process (reducing the number of possible combinations for the subsequent selection process) to estimate the solutions, which has enabled scoring and rank ordering of combinations of components. The proposed framework and techniques has been illustrated and demonstrated for a robotic vacuum cleaner (RVAC) design example. The proposed filtering process in the RAVC design example has eliminated 8 incompatible combinations out of a total 16 combinations, and has determined combination(11) with HEPA, Type A, Li-ion, and Dust bag or plastic bin as the best compatible combination.

The proposed compatible combinations having the largest intersection with the feasible space under the design constraints (i.e. having the maximum number of feasible tuples) will provide design teams with flexibility in controlling the design variables for further optimisation in the subsequent design stages.

6. REFERENCES

1. Bailleux, O., and Chabrier, J., 1997, "Counting by Statistics on Search Trees: Application to Constraint Satisfaction Problems," Intelligent Data Analysis 1, pp. 263-273.
2. Benhamou, F., Goualard, F., Granvilliers, F., and Puget, J.-F., 1999, "Revising Hull and Box Consistency," Proceedings of the International Conference on Logic Programming, pp. 230-244.
3. Bolloju, N., Christoph, S., and Vijayan, S., 2012, "A Knowledge-based System for Improving the Consistency between Object Models and Use Case Narratives", Expert Systems with Applications, 39, pp. 9398-9410.
4. Bradley, S. R., and Agogino, A. M., 1994, "An Intelligent Real Time Design Methodology for Component Selection: An Approach to Managing Uncertainty," Journal of Mechanical Design, 116, pp. 980-988.
5. Carlson-Skalak, S., 1996, "Genetic Algorithm Attributes for Component Selection," Research in Engineering Design, 8, pp. 33-51.
6. Darr, T., and Birmingham, W., 2000, "Part-Selection Triptych: A Representation Problem Properties and Problem Definition, and Problem-Solving Method," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 14(1), pp. 39-51.
7. Dechter, R., and Pearl, J., 1988, "Network-Based Heuristics for Constraint-Satisfaction Problems,"

Artificial Intelligence, 34, pp. 1-38.
8. Director, S. W., and Hachtel, G. D., 1977, "The Simplicial Approximation Approach to Design Centering," IEEE Transactions on Circuits and Systems, CAS-24(7), pp. 363-371.
9. D'ambrosio, J., and Birmingham, W., 1995, "Preference-Directed Design," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 9, pp. 219-230.
10. Faltings, B., 1994, "Arc-Consistency for Continuous Variables," Artificial intelligence, 65(2), pp. 363-376.
11. Freuder, E. C., Wallace, R. J., and Heffernan, R., 2003, "Ordinal Constraint Satisfaction," 5th International Workshop on Soft Constraints-Soft 2003, County Cork, Ireland.
12. Granvilliers, L., and Benhamou, F., 2006, "Algorithm 852: RealPaver: An Interval Solver Using Constraint Satisfaction Techniques," ACM Transactions on Mathematical Software, 32(1), pp. 138-156.
13. Han, C., and Lee, C., 1988, "Comments on Mohr and Henderson's Path Consistency Algorithm," Artificial Intelligence, 36, pp. 125-130.
14. Junker, U., 2002, "Preference Based Search and Multi Criteria Optimization," Proceedings of the18th National Conference on Artificial Intelligence, Edmonton, Alberta, Canada, pp. 34-40.
15. Karni, R., and Belikoff, S., 1996, "Concurrent Engineering Design Using Interval Methods," International Transactions in Operational Research 3(1), pp. 77-87.
16. Kim, D., Bufardi, A., and Xirouchakis, P., 2006, "Compatibility Measurement in Collaborative Conceptual Design," Annals of the CIRP, 55(1), pp. 151-154.
17. Lhomme, O., 1993, "Consistency Techniques for Numeric CSPs," Proceedings of the International Joint Conference on Artificial Intelligence, pp. 232-238.
18. Lottaz, C., Smith, I., Robert-Nicoud, Y., and Faltings, B., 2000, "Constraint-Based Support for Negotiation in Collaborative Design," Artificial Intelligence in Engineering, 14(3), pp. 261-280.
19. Mackworth, A., 1977, "Consistency in Networks of Relations," Artificial Intelligence, 8(1), pp. 99-118.
20. Medland, A. J., and Mullineux, G., 2000, "A Decomposition Strategy for Conceptual Design," Journal of Engineering Design, 11(1), pp. 3-16.
21. Mittal, S., and Falkenhainer, B., 1990, "Dynamic Constraint Satisfaction Problems," Proceedings of the 8th AAAI, Boston, pp. 25-32.
22. Nadel, B., 1989, "Constraint Satisfaction Algorithms," Computational Intelligence, 5, pp. 188-224.
23. Neumaier, A., 1990, Interval Methods for Systems of Equations, Cambridge University press, Cambridge.
24. Pahl, G., and Beitz, W., 1988, Engineering Design - a Systematic Approach, Springer-Verlag, London.
25. Patel, M., Lewis, K., Maria, A., and Messac, A., 2003, "System Design through Subsystem Selection Using Physical Programming," AIAA Journal, 41, pp. 1089-1096.
26. Qureshi, A. J., Jeans, Y. D., Jerome, B., and Regis, B., 2010, “Set Based Robust Design of Mechanical Systems using the Quantifier Constraint Satisfaction Algorithm,” Engineering Application of Artificial Intelligence, 23, pp. 1173-1186.
27. Roth, D., 1996, "On the Hardness of Approximate Reasoning," Artificial Intelligence, 82, pp. 273-302.
28. Sabin, D., and Freuder, E. C., 1996, "Configuration as Composite Constraint Satisfaction," Proceedings of the Artificial Intelligence and Manufacturing Research Planning Workshop, pp. 153-161.
29. Sam-Haroud, D., and Faltings, B., 1996, "Consistency Techniques for Continuous Constraints," Constraints, 1, pp. 85-118.
30. Singhal, J., and Singhal, K., 1996, "The Number of Feasible Designs in a Compatibility Matrix," European Journal of Operational Research, 94, pp. 186-193.
31. Tsang, E., 1993, Foundations of Constraint Satisfaction, Academic Press, San Diego.
32. Yao, Z., and Johnson, A. L., 1997, "On Estimating the Feasible Solution Space of Design," Computer-Aided Design, 29(9), pp. 649-655.