Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1999), *13*, 205–215. Printed in the USA.

A selective, multiple-criteria method for handling constraint violations in well-defined design problems

ARGIRIS J. DENTSORAS

Machine Design Laboratory, Department of Mechanical Engineering & Aeronautics, University of Patras, 26500, Patras, Greece

(Received October 22, 1997; Revised October 15, 1998; Accepted February 25, 1999)

Abstract

A selective, multiple-criteria method is proposed for handling constraint violations in well-defined design problems. First, the types of design problems upon which the method may apply are presented and a graph topology is adopted for representing the problem decomposition in the level of design parameter interrelations (design parameter graphs). Next, the problem of constraint violations for the design parameters is discussed. It is shown that these violations can be resolved by modifying the values of the primary design parameters and recalculating the values of the violated parameters. Any "blind" attempt of modifying the values of the primary design parameters for resolving occurring violations may, eventually, create additional violations. The proposed method guides the designer toward those primary design parameters that present the least possibility of creating more violations when their values are modified. This is achieved by applying multiple criteria and by producing a final, sorted list of primary design parameters. The designer may then choose the first element of this list to handle efficiently the violations of the design parameters. Two examples are given on a design space where constraint violations occur. Through these examples, the capability of the proposed method in helping the designer to handle constraint violation is shown. The concluding remarks, except for summarizing the potential of the method, determine its boundaries and include a reference on relative work currently under investigation.

Keywords: Design; Design Parameters; Constraints; Violation Handling

1. INTRODUCTION

The design of products is an integrated engineering activity that, during the last decades, has become the object of intensive academic and applied research. This is due to the fact that design has been widely accepted as a "sine que non" process for the manufacturing of marketable, high-quality products.

While design contributes toward products of high final quality, the theory of design itself is still a discipline viewed as a "soft," rather than well-defined subject. According to Dym (1994), this is mainly because it is not sufficiently mathematical and does not occupy a rigorous common vocabulary for analyzing and describing itself as well as the products designed. Therefore, during the last years, and in parallel with solving specific design problems, research efforts have focused toward establishing axioms, theories, methods, techniques, and a terminology vocabulary that could apply—with minor modifications—to a wide spectrum of design problems.

Two major problems in the theory of design are the formal representation of the available design knowledge and its exploitation for producing feasible design solutions. Pure mathematical tools and conventional computer-aided design (CAD) environments, are only partially able to provide solutions for the above problems because they can only in part represent and handle nonquantitative, empirical, and ill-structured knowledge. Recent achievements in the field of Artificial Intelligence offer the opportunity of overcoming—to a certain extent—these problems by providing methods and techniques that can capture, organize, and exploit the available design knowledge to provide better design solutions.

In design, a decomposition of the design problem is always required. This decomposition may be implemented in different abstraction levels and under different aspects of

Reprint requests to: Argiris J. Dentsoras, Machine Design Laboratory, Department of Mechanical Engineering & Aeronautics, University of Patras, 26500 Patras, Greece. E-mail: dentsora@mech.upatras.gr

view (Banares-Alcantara, 1991). More specific, someone may decompose:

- the designed product into its parts;
- the design process into its discrete successive steps;
- the design problem into subproblems (Dym, 1994); and
- the design parameters into simpler parameters (Dentsoras, 1996; Tsalides and Dentsoras, 1997), etc.

While each time the type of decomposition depends upon its scope, there must always be a valid method for its implementation in a formal and robust way.

In every design problem there are always design constraints. There are different types of constraints (functional, topological, material, etc.) that usually refer to different design issues and may appear during almost all the stages of the design process (Cross, 1991; Serrano & Gossard, 1992; Ullman, 1992).

For the majority of the design problems, a large number of design parameters and even a larger number of constraints are necessary for the satisfactory representation of the inherent design knowledge. Concerning constraints, their representation and handling during design are considered difficult problems and various methods and techniques have been proposed for their solution.

In the present paper, a selective, multiple-criteria method is presented for resolving constraint violations in well-defined design problems. The method uses directed graph structures for the representation of the design parameters and does not require the construction of constraint networks. Because one or more violations may occur for one or more design parameters, it is shown that these violations can be resolved by modifying the values of some primary design parameters and by recalculating, subsequently, the values of the violated parameters.

Any "blind" attempt of modifying the values of the primary design parameters for resolving the violations may, eventually, create additional violations. The method guides the designer toward only those primary design parameters that are related to the violated design parameter and present the least possibility of creating additional violations when their values change. For any violated parameter, multiple successive criteria apply which, through a systematic search on the design parameter graph structures, produce a sorted list of primary design parameters. Then, the designer may choose a primary design parameter from this list and try to resolve efficiently the violation through repetitive value assignments to this primary design parameter accompanied by propagation of its value.

The proposed method does not aim in ensuring that there will always be a solution to the problem of constraint violations for a design problem. Its main concern is to avoid the generation of more violations and help the designer to handle efficiently the occurring violation. The program code for the implementation of the method was written in Microsoft Visual Basic version 5 and developed as a module of an already existing design environment EXTDSEARCH (Dentsoras, 1996; Tsalides & Dentsoras, 1997) used for solving well-defined design problems.

2. WELL-DEFINED DESIGN PROBLEMS—DESIGN CONSTRAINTS

2.1. Well-defined design problems

According to Dym (1994), design problems are typically open-ended because they usually present multiple acceptable solutions. They are also ill-structured because the solution-finding process—for most of cases—is not routine and cannot be represented by a series of mathematical formulae and rules. However, there are many design cases based upon a critical amount of well-established design knowledge repeatedly tested in everyday design practice. This knowledge can be expressed as formulae and rule sets that produce acceptable solutions. This is the case mostly for routine problems (see below). These problems are considered as *well-defined* and, as a consequence, the representation of the corresponding knowledge is much easier than for the case of ill-structured problems. Cross (1991), additionally, considers well-defined design problems having often only one correct answer (solution).

During the last years, there has been a large research effort toward establishing a widely accepted taxonomy for the various kinds of design problems. Although this work has not yet been accomplished, there is a consensus (Dym, 1994; Tong & Sriram, 1992; Ullman, 1992) that any design problem can be classified under one—or a combination—of the following types: *Selection design, configuration design, parametric design, original design, redesign, and routine design*. (Some authors use different names to describe the same type of design problem.)

The above classification of the design problems is based on differences concerning various aspects and characteristics of the design problem under consideration. For example, in original design, a lot of creative work has to be done during the stage of conceptual design to produce new ideas. These will be later elaborated—during the stage of detailed design—to produce final solutions. In contrast, for the problems of the routine type and for the most of the cases, the stage of the conceptual design is almost absent. The design process presents the form of a well-established procedure that, when followed strictly, will produce feasible design solutions.

Routine design is the most typical representative of well-defined design. In routine design, the inherent knowledge can be represented and handled easier than for other ill-structured types of problems. This fact does not retract, however, any possible complexity of the problem itself and does not reduce the effort for introducing better representation schemes and establishing better knowledge handling strategies for its solution.

In the present work, the domain of interest is restricted to well-defined, routine design problems. It is assumed that

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

design is performed in nonautomated, designer-defined manner that allows value assignments only to primary design parameters (see Section 3.1). The knowledge of the problem can be expressed by formulae, rule sets, and databases. For the representation of the design knowledge, it is assumed that its decomposition can be implemented on the following levels (Dentsoras, 1996):

- problem (tasks or subproblems) and
- parameters (derived and primary).

The design parameters and the design tasks are represented by frame-like structures (Dentsoras, 1996). Their interrelations are expressed formally through design parameter graph structures (see next section). Then, a simple depth-first search method is applied on those structures and feasible solutions are generated. On these solutions, multiple designer-defined criteria may apply which, through a filtering process, will produce the final solutions (Dentsoras, 1996).

2.2. Design constraints

In every design problem, there are always constraints that restrict the number of its final acceptable solutions. These constraints can either appear as initial requirements during the stage of the problem's statement or be produced during the design process as a result of the decisions made by the designer (Ullman, 1992).

Many design problems are considered as constraint-satisfaction problems (CSP) because their solution depends on the ability to articulate, apply, and satisfy the constraints that form the implicit statement of the design goals (Dym, 1994). The constraints comprise a part of the knowledge about the problem itself and they must be represented and handled together with the rest of the knowledge in every stage of the design process. Various models have been proposed for their representation and handling.

There are several methods that may apply for the solution of a CSP. The most common is the generate-and-test method, where each possible combination of the parameters is systematically generated and tested to see whether it satisfies all the constraints (Kumar, 1992). A more effective method is backtracking where as soon as all the parameters relevant to a constraint are instantiated, the validity of the constraint is checked. Then, in case a violation occurs, backtracking is performed to the most recently instantiated parameter (Kumar, 1992).

A problem connected with the backtracking method is the consideration of the order for the instantiation of the parameters. According to the search rearrangement method proposed by Bitner and Reingold (1975), the parameter with the fewest possible remaining alternatives is selected for instantiation. The instantiation order is different in different branches of the constraint tree and is, generally, dynamically determined. Freuder and Quinn (1985) proposed a heuristic method according to which these parameters are instantiated first that participate in the highest number of constraints. This approach results in early pruning of the nonsuccessful branches of the constraint tree.

According to a paper by Brown (1985), constraint failures (violations) are the most common failures occurring during the design process. Because the problem is always the proper handling of these failures, failure handlers and redesigners are introduced in this paper together with a presentation of action and knowledge concerning failure recovery.

Brown and Chandrasekaran (1992) investigate the knowledge and control structures that characterize design as a generic problem-solving activity. They consider constraints as agents that test for interrelations between attributes at any stage of the design process. One kind of failure that may occur during design is constraint violation. This violation is handled as a demand for redesign and is treated as soon as it occurs by proper modules, namely Failure Handler and Redesigner.

In one of their early papers, Serrano and Gossard (1988) present a constraint-based environment for Mechanical CAD that can be used for the early stages of design. The management of the constraints is implemented through constraint networks of directed graphs. The authors also present techniques for the evaluation of constraint networks, the detection of over- and underconstrained systems of constraints, as well as for the identification and correction of redundant and conflicting constraints. Constraint management is implemented in the concept modeler—a system for conceptual design—and is based on the causal-dependency sphere metaphor introduced by the authors. In a later work of the same authors (Serrano & Gossard, 1992), reference is made to sensitivity analysis concerning the interdependency between design parameters. They conclude that this sensitivity analysis may be used for the constraint system too. They present also a short mathematical analysis that covers the subject of the estimation of the requested sensitivity of a certain design parameter.

Taylor and Corlett (1993) present a knowledge-based approach to CAD and propose a methodology based on the successive refinement of the design constraints implemented by the ALFIE expert system.

A framework introduced by Mittal and Araya (1992) and based partly upon the expert system PRIDE, applies to well-defined design spaces and organizes the relevant knowledge as design plans. The authors emphasize the fact that, for such kinds of design spaces, the process of generating alternative designs is largely a process of searching these spaces. They also describe a problem-solver that executes the design plans and uses information produced from a constraint violation as feedback for modifying a partial design.

Nadel and Lin (1992) consider the automation of automobile transmission design as a constraint satisfaction problem that involves three components: variables, values, and constraints. The problem is represented by corresponding sets of these components and Cartesian products. It is then

reduced in finding value-tuples in the overall Cartesian product.

Marcus et al. (1988) presented the expert elevator designer VT that was based on the edge-acquisition tool SALT developed by the authors. VT uses the strategy of constructing a plausible approximation and then successfully refining it through extensive backtracking search on the constraint graphs.

In the present paper, design constraints are considered as attributes of the design parameters and are embodied as attribute slots in their representation frames. The test for constraint violations is performed in parallel with the assignment of values to the parameters.

The method described below is based on the fact that any violations of the design constraints can be resolved by modifying the values of the primary design parameters that the designer has full access to and by recalculating the problematic parameters according to these modified primary parameters. In this context, the problem solved is not a CSP because no method is provided for obtaining the modified set of primary design parameter values that would satisfy all constraints and resolve the constraint violation. In contrast, when applied in constraint violation cases, the method helps the designer to handle the primary design parameters in a way that minimizes the possibility of producing new violations. Below, an analytical presentation of this method is given.

3. A SELECTIVE, MULTIPLE—CRITERIA METHOD FOR RESOLVING CONSTRAINT VIOLATION

3.1. Introduction

In every well-defined design problem, a minimum number of design parameters are required to express the built-in design knowledge. The *determinative interrelations* (calculation formulas, expressions, procedures, rules, etc.) among these parameters can be represented *by design parameter graph structures* (DPGS). Every design parameter can be expressed by a respective DPGS so that the number of DPGSs is always equal to the total number of the design parameters.

A design parameter graph structure is:

1. *connected* (Bose & Ellacot, 1991) because there is at least one path between every pair of design parameters and

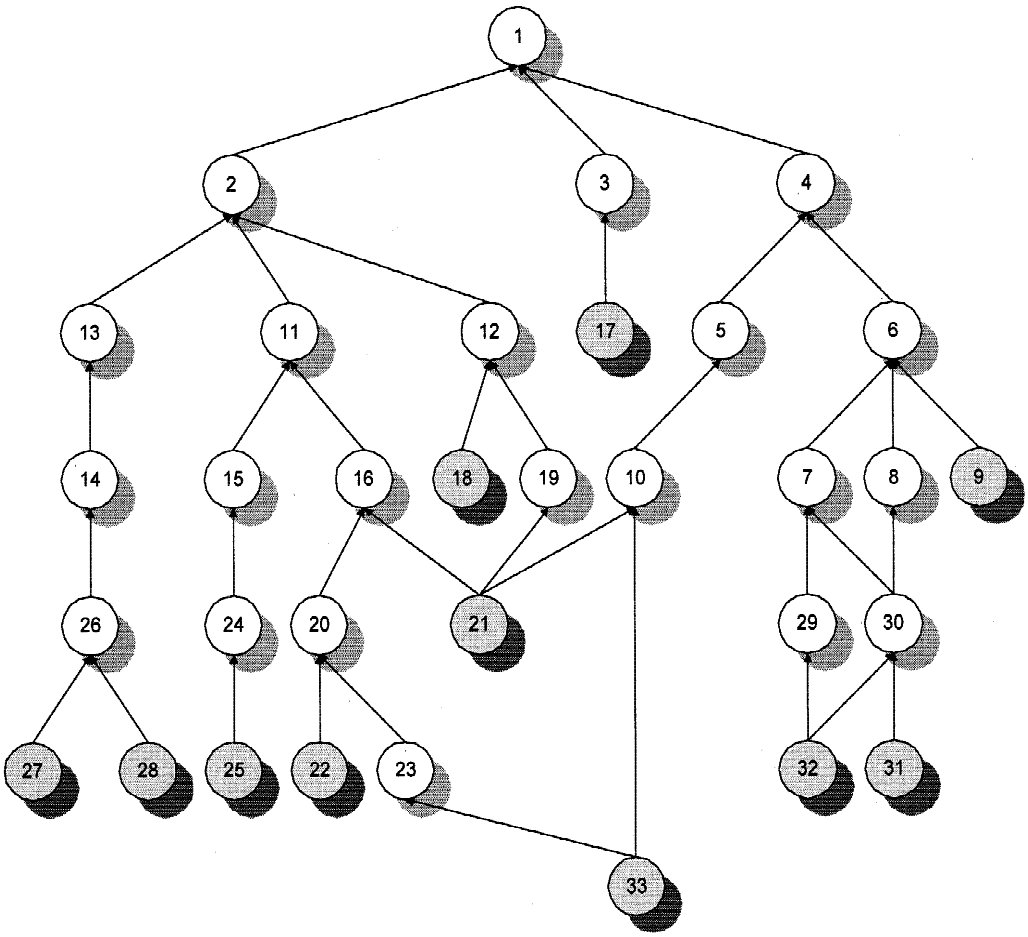

Fig. 1. Schematic representation of a DPGS for design parameter 1. The grey-coloured parameters are primary.

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

2. *directed* (Bose & Ellacot, 1991), always starting from its leaves (*primary or independent design parameters)* toward its root (*root design parameter or simply design parameter)* through one or more *derived design parameters*.

As an example, a schematic representation of a DPGS for design parameter 1 (root design parameter) is shown in Figure 1. Its value arises from the values of the design parameters in the lower level. For each one of those parameters, a unique DPGS can also be drawn.

Any DPGS can be considered as a *union* of one or more DPGS. This can be expressed as follows:

IF: $G_1 = (D_1, C_1)$ is a DPGS that consists of a set of design parameters:

$$D_1 = \{d_1, d_2, \ldots, d_d\}, d \geq 1$$

and a set of relations between them:

$$C_1 = \{c_1, c_2, \ldots, c_d\}, d \geq 0$$

and $G_2 = (D_2, C_2)$ is another DPGS that consists of the set of design parameters D_2 (not necessarily different from D_1) and the set of relations C_2 (not necessarily different from C_1),

THEN: the *union* of these two graphs is also a DPGS graph $G_3 = (D_3, C_3)$:

$$G_3 = G_1 \cup G_2 \text{ with } D_3 = D_1 \cup D_2 \text{ and } C_3 = C_1 \cup C_2.$$

For the case that D_i has only one element, then C_i is mandatory an empty set (index $c = 0$). As a consequence, the corresponding DPGS contains one vertex and no relations. This is the case for the *G* sets of the primary design parameters only.

According to the proposed approach, to assign a value to a certain design parameter, it is necessary to construct its DPGS from other, simpler DPGSs (synthesis of several DPGS) by applying recursive procedures. For the generation of the set of primary design parameters in a certain DPGS, a reverse process (analysis) is needed, which can also be implemented by applying recursive procedures. In both cases, a subset is used of the set containing all the design parameters of the problem under consideration.

Depth-first method (Rich & Knight, 1991) is used as a recursive search procedure for synthesis and analysis. The use of this exhaustive technique is necessary because no heuristic knowledge is provided concerning the relationships among the design parameters. By applying this method, the DPGS for any design parameter can be constructed easily and all the primary design parameters in this DPGS can be located easily. The depth of a DPGS for a certain design parameter varies according to the amount of its interrelations with other parameters. This depth is equal to zero only for primary design parameters. Therefore, the complexity of the algorithm for constructing DPGSs and locating primary design parameters depends upon the design problem under consideration. More specific, it depends upon the decomposition into different tasks and subtasks and it can not be determined *a priori*.

Consider the design parameter d_v and its DPGS $G_v = (D_v, C_v)$. For this parameter, it is further assumed that a violation has occurred. By applying the depth-first method, a list L_v of primary design parameters can be formed. For example, if design parameter 4 has been violated (see Fig. 2), then the corresponding list will be $L_4 = 21,33,32,31,9$. The *D*-sets of these primary parameters have only one member and their *C*-sets are empty.

Any value change for any of the parameters in the list L_v may resolve the violation for the parameter d_v. Then, *through a recursive synthetic process (see above), the value of d_v will be recalculated and may be found within the acceptable limits.* Therefore, the designer may eventually overcome the violation of d_v through repetitive value assignments to the parameters of the L_v list.

While trying to resolve the violation for d_v by changing the values of the parameters of the list L_v, there is always the possibility of generating additional violations to other parameters' constraints. So, there must be a method that would suggest to the designer to use that primary parameter from the list L_v that presents the least side effects for the rest nonprimary parameters. The above problem can be solved through a rearrangement of the list L_v in ascending

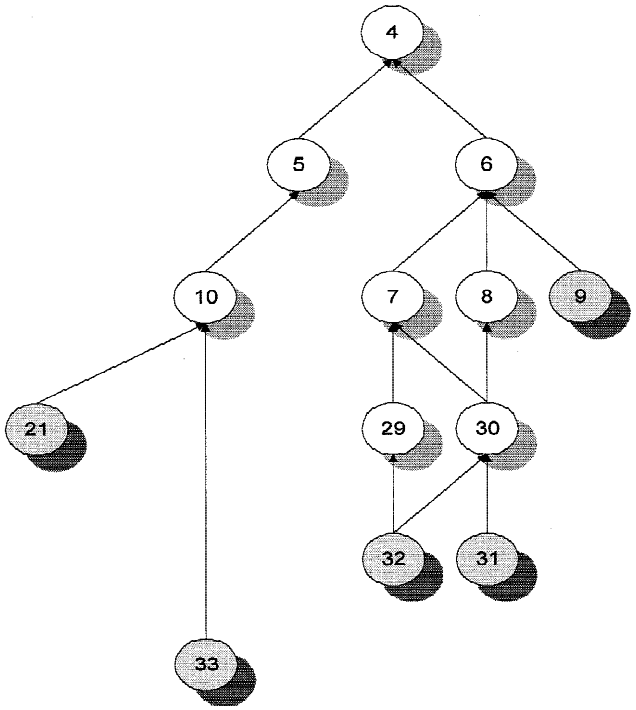

Fig. 2. Schematic representation of a DPGS for design parameter 4. The grey-coloured parameters are primary. (The above DPGS is a subpart of the DPGS for design parameter 1, see Fig. 1)

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

order. Then the first element of this list is the primary design parameter that will produce the least side effects when its value is changed to produce a new value for d_v.

For the arrangement of the elements of the list, four criteria are proposed. These criteria are then combined to comprise an integrated method. Additionally, weight coefficients are used to produce the final sorted list. The detailed analysis is given below.

3.2. Criterion 1: Distance from the violated parameter

The distance of a primary design parameter in a DPGS is equal to the number of the arcs connecting the vertices along any path starting from it and ending to the root vertex. According to this criterion, the shorter the distance of the primary design parameter from the violated parameter along a path of its DPGS, the smaller the possibility of producing additional violations when its value changes. Indeed, any intermediate vertex (parameter) in this path has a determinative relation with the primary design parameter and is a candidate for possible violation. So the less the intermediate vertices, the smaller will be the possibility of producing additional violations.

By calculating all the distances of the members of the L_v from d_v, the original L_v can be rearranged in an ascending order. Because there may be more than one paths from a primary parameter to a violated parameter and, therefore, more than one distances, the maximum of them is always considered.

As an example, suppose that:

$$L_v = d_1, d_2, d_3$$

and:

$$dist(d_1, d_v) = 3$$

$$dist(d_2, d_v) = max\{dist(d_2, d_v)_1, dist(d_2, d_v)_2\} = max\{3,4\} = 4$$

$$\text{dist}(d_3, d_v) = 1,$$

where there are two paths from d_2 to d_v. Then the list L_v will be rearranged as:

$$L_1 = d_3, d_1, d_2.$$

3.3. Criterion 2: Distance from the design parameter of the maximum *G*-set

In every design problem, there is always a design parameter with the maximum *G*-set. The DPGS of this parameter contains the largest number of design parameters when compared with the DPGSs of the rest parameters. The DPGS of this parameter may either contain the violated design parameter, which implies that contains all the parameters of its L_v, or one or more of the parameters in the L_v. In every case, the distance among each primary parameter in the L_v and the design parameter with the maximum *G*-set must be determined and the original L_v must then be rearranged in an ascending order. As an example, suppose that:

$$L_v = d_1, d_2, d_3$$

and d_g is the parameter with the maximum *G*-set. By assuming that the DPGS d_g contains all the parameters in L_v, the corresponding distances are:

$$dist(d_1, d_g) = 6$$

$$dist(d_2, d_g) = max\{dist(d_2, d_g)_1, dist(d_2, d_g)_2\} = max\{5,6\} = 6$$

$$dist(d_3, d_g) = max\{dist(d_3, d_g)_1, dist(d_3, d_g)_2\} = max\{2,1\} = 2$$

where there are two paths from both d_2, d_3 to d_g. Then the list L_v will be rearranged as:

$$L_2 = \underline{d_3, d_2}, d_1.$$

The underlined elements in L_2 present the same distance from d_g.

3.4. Criterion 3: Number of parent design parameters

A primary design parameter connected to a number of parent parameters is more suspicious for producing additional constraint violations than one with a smaller number of parent parameters. So, an estimation is done of the number of the parent design parameters for each member of the original L_v list. The list is then rearranged in an ascending order. As an example, suppose that:

$$L_v = d_1, d_2, d_3$$

and:

$$par(d_1) = 3,$$

$$par(d_2) = 1, \text{ and}$$

$$par(d_3) = 1.$$

Then the list L_v will be rearranged as:

$$L_3 = \underline{d_2, d_3}, d_1.$$

The underlined elements of L_3 present the same number of parent parameters.

3.5. Criterion 4: Analytical determination of affected parameters

According to this criterion, a detailed calculation is performed of the number of all the affected parameters for each primary design parameter in the original list L_v and the corresponding lists are created. Then, depending on the number of the elements of each list, the original L_v list is

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

rearranged in an ascending order. As an example, suppose that:

$$L_v = d_1, d_2, d_3$$

and:

$$al(d_1) = d_4, d_7, d_{11}$$

$$al(d_2) = d_6, d_9, d_{11}, d_{13}, d_{15}$$

$$al(d_3) = d_5, d_8, d_9, d_{10}, d_{12}, d_{17}.$$

Then the list L_v will be rearranged as:

$$L_4 = d_1, d_2, d_3.$$

3.6. Combination of the criteria—The final list

In certain simple design cases, the above four criteria may be applied individually to produce a sorted list of primary design parameters. However, for most of the cases, the lists produced by the application of the above four criteria must be further submitted to a final elaboration to produce the final sorted list. This list will be a product of the combined action of those criteria and will also incorporate the influence of weight coefficients as follows:

Concerning the impact of each primary parameter's position in the list, the weight coefficient of the first position will be greater than the weight coefficient of the second etc. Then:

IF: p is the position index of a primary parameter d in a list L,

THEN: $p = 1,2,\ldots,n$, where $n \geq 1$ is the number of the primary parameters in the list L_v,

L_i, $i = 1,2,3,4$ is the list produced by the application of i-criterion,

$(f_{d,p})_{L_i}$ = appearance of d in p-position in the L_i list (its value can be either 0 or 1),

$f_{d,p} = \sum_{i=1}^{4} (f_{d,p})_{L_i}$ = frequency of appearance of d in p-position for all L_i lists and: $0 \leq f_{d,p} \leq 4$,

w_p = weight coefficient of p-position (usually $w_p = n - p + 1$, because the first position in the list is more significant that second position etc.). Although the designer may choose another formula or way to define the values of the weight coefficients, s/he should take into account that the ith position in a list is always more significant than the $(i-1)$th.

The final score for each primary parameter d in the L_f list will be given by the relation:

$$r_d = \sum_{p\text{-}1}^{n} (w_p f_{d,p}) = \sum_{p=1}^{n} \left\{ (n - 1 + p) \sum_{i=1}^{4} (f_{d,p})_{L_i} \right\}.$$

By applying the above relation for every primary parameter in the list, a final ranking among them is obtained. Then the designer can choose the parameter with the highest score and assign a new value to it to force recalculation of the violated design parameter.

For the case that the scores of two or more primary parameters for a certain position in the final list L_f are equal, the problem can be solved by taking into account the frequencies of appearance of these parameters for this position.

Many physical and engineering problems are well-defined design problems and can be represented by suitable design parameter graph structures similar to that shown in Figure 1. Recently, Tsalides and Dentsoras (1997) used design parameter graph structures to perform design of a belt conveyor. The DPGS shown in Figure 1 was extracted from this design case and it refers to the determination of the conveyor belt width. The original parameter names were replaced by numbers to facilitate the analysis in the present paper. The two examples given below were extracted from the DPGS shown in Figure 1.

3.7. Examples

Consider parameter 4 (see Fig. 1) as the violated parameter. Its DPGS is shown in Figure 2.

After applying the depth-first method, the original list of the primary design parameters is found to be:

$$L_v = 21,33,32,31,9.$$

Next, the four criteria apply sequentially. The resulting lists are:

1. $dist(21,4) = 3$,

   $dist(33,4) = 3$,

   $dist(32,4) = max\{dist(32,4)_1, dist(32,4)_2\} = max\{4,4\} = 4$,

   $dist(31,4) = max\{dist(31,4)_1, dist(31,4)_2\} = max\{4,4\} = 4$,

   $dist(9,4) = 2$

   and: $L_1 = 9,\underline{21,33},\underline{32,31}$.

   The underlined elements in L_1 present the same distance.

2. $dist(21,1) = max\{dist(21,1)_1, dist(21,1)_2, dist(21,1)_3\} = max\{4,4,4\} = 4$

   $dist(33,1) = max\{dist(33,1)_1, dist(33,1)_2\} = max\{6,4\} = 6$,

   $dist(32,1) = max\{dist(32,1)_1, dist(32,1)_2\} = max\{5,5\} = 5$,

   $dist(31,1) = max\{dist(31,1)_1, dist(31,1)_2\} = max\{5,5\} = 5$

   $dist(9,1) = 3$

   and: $L_2 = 9,21,\underline{31,32},33$.

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

The underlined elements in L_2 present the same distance.

3. $par(21) = 3$, $par(33) = 2$, $par(32) = 2$, $par(31) = 1$,

   $par(9) = 1$

   and: $L_3 = \underline{9,31},\underline{32,33},21.$

   The underlined elements in L_3 present the same number of parent parameters.

4. $al(21) = 16,11,2,1,19,12,10,5,4$

   $al(33) = 23,20,16,11,2,1,10,5,4$

   $al(32) = 29,7,6,4,1,30,8$

   $al(31) = 30,7,6,4,1,8$

   $al(9) = 6,4,1$

   and: $L_4 = 9,31,32,\underline{33,21}$.

The underlined elements in L_4 present the same number of affected parameters. Lists L_1–L_4 form the basis for the production of a final sorted list. For every primary d parameter of the list L_v and according to its position in this list, its appearance in every L_i list and for every position p is recorded. Then, for parameter 21:

$p = 1$: $i = 1$: $(f_{21,1})_{L_1} = 0$

$i = 2$: $(f_{21,1})_{L_2} = 0$

$i = 3$: $(f_{21,1})_{L_3} = 0$

$i = 4$: $(f_{21,1})_L 4 = 0$ and finally: $f_{21,1} = \sum_{i=1}^{4} (f_{21,1})_{Li} = 0$

$p = 2$: $i = 1$: $(f_{21,2})_{L_1} = 1$

$i = 2$: $(f_{21,2})_{L_2} = 1$

$i = 3$: $(f_{21,2})_{L_3} = 0$

$i = 4$: $(f_{21,2})_{L_4} = 0$ and finally: $f_{21,2} = \sum_{i=1}^{4} (f_{21,2})_{Li} = 2$

$p = 3$: $i = 1$: $(f_{21,3})_{L_1} = 1$ (same distance with 33)

$i = 2$: $(f_{21,3})_{L_2} = 0$

$i = 3$: $(f_{21,3})_{L_3} = 0$

$i = 4$: $(f_{21,3})_{L_4} = 0$ and finally: $f_{21,3} = \sum_{i=1}^{4} (f_{21,3})_{Li} = 1$

$p = 4$: $i = 1$: $(f_{21,4})_{L_1} = 0$

$i = 2$: $(f_{21,4})_{L_2} = 0$

$i = 3$: $(f_{21,4})_{L_3} = 0$

$i = 4$: $(f_{21,4})_{L_4} = 1$ (same number of affected parameters with 33)

and finally: $f_{21,4} = \sum_{i=1}^{4} (f_{21,4})_{Li} = 1$

$p = 5$: $i = 1$: $(f_{21,5})_{L_1} = 0$

$i = 2$: $(f_{21,5})_{L_2} = 0$

$i = 3$: $(f_{21,5})_{L_3} = 1$

$i = 4$: $(f_{21,5})_{L_4} = 1$ (same number of affected parameters with 33)

and finally: $f_{21,5} = \sum_{i=1}^{4} (f_{21,5})_{Li} = 2.$

The frequencies of appearance for parameter 21 are shown in Table 1, together with the frequencies of the rest parameters in the list L_v. The table contains also the position weight coefficients and the final scores.

The first row of Table 1 contains the position indices in the lists. The second contains the corresponding weight coefficients. Then position 1 corresponds to the greatest weight coefficient while position 5 to the smallest one. In the lower part of the table, the frequency of appearance of each primary parameter in each position is given. For example, the frequency of appearance of parameter 33 in position 3 for all L_i lists is 2. Finally, the right column in the table contains the final scores for each primary parameter. For example, the score for parameter 9 is calculated as follows:

$$r_9 = \sum_{p=1}^{5} (w_p f_{9,p}) = \sum_{p=1}^{5} \left\{ (5 - 1 + p) \sum_{i=1}^{4} (f_{9,p})_{Li} \right\}$$

$$= 5 \cdot 4 + 4 \cdot 1 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 0 = 24.$$

According to the scores in the right column, the final list will be: $L_f = 9,31,\underline{33,32},21$. The equal scores between the parameters 33 and 32 for the 3rd position in L_f causes a minor problem that can be easily solved by comparing their frequencies for that position. Because $\left(\sum_{i=1}^{4} (f_{33,3})_{Li} = 2\right) \leq \left(\sum_{i=1}^{4} (f_{32,3})_{Li} = 3\right)$, parameter 32 presents a greater frequency for that position in all L_i lists and finally the L_f becomes:

$$L_f = 9,31,32,33,21.$$

The designer may choose any of the primary parameters contained in this list to change its value and recalculate the value

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

Table 1. *Frequencies of appearance and final scores for the violated parameter 4*

Position in list s $p =$	1	2	3	4	5	
Weight coef. of p-position $w_p =$	5	4	3	2	1	Final score $\sum_{p=1}^{n} (w_p f_{d,p})$
Parameter d	Total frequency of appearance $f_{d,p}$ in p-position $\sum_{i=1}^{4} (f_{d,p})_{L_i}$					
21	0	2	1	1	2	15
33	0	1	2	2	2	16
32	0	0	3	3	1	16
31	1	2	1	1	1	19
9	4	1	0	0	0	24

of the violated parameter. However, by choosing the first one, the danger of producing additional violations to other nonprimary parameters is kept to a minimum according to the analysis concerning the application of the four criteria.

For completing the presentation of the method, another example (see Table 2) is given concerning the violation of parameter 11 (see Fig. 3 for its DPGS). The final list L_f is:

$$L_f = \underline{25,22},21,33.$$

Because there is no difference between the frequencies of parameters 25, 22 for the first position to resolve the equality problem of the corresponding scores in L_f, the designer can either choose 25 or 22 as the parameter with the least danger for producing more violations.

The proposed method can be easily programmed through the extensive use of recursive procedures. By optimizing the program code, the computation time is kept within reasonable limits. A general outline of the list generation procedure is shown in Figure 4.

4. CONCLUDING REMARKS

In the present paper, a selective, multiple-criteria method is presented for handling constraint violations in well-defined design problems. The proposed approach is well-suited for cases of well-defined design spaces where all the design knowledge is known *a priori* and can be configured as graph structures in various decomposition levels. The method acts as an assistance tool for the designer because it suggests

Table 2. *Frequencies of appearance and final scores for the violated parameter 11*

Position in list s $p =$	1	2	3	4	
Weight coef. of p-position $w_p =$	4	3	2	1	Final score $\sum_{p=1}^{n} (w_p f_{d,p})$
Parameter d	Total frequency of appearance $f_{d,p}$ in p-position $\sum_{i=1}^{4} (f_{d,p})_{L_i}$				
25	2	4	2	0	24
22	2	4	2	0	24
33	0	0	2	3	7
21	2	0	0	2	10

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

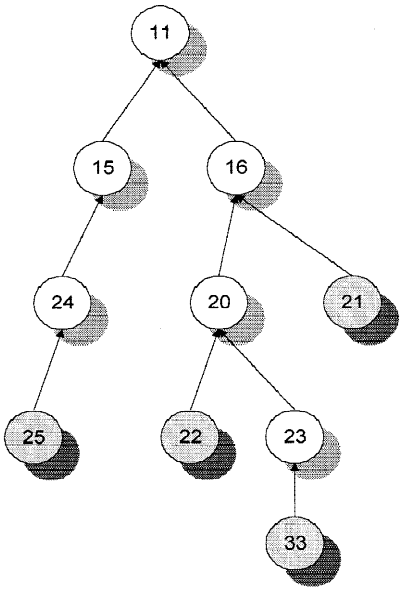

Fig. 3. Schematic representation of a DPGS for design parameter 11. The grey-coloured parameters are primary. (The above DPGS is a subpart of the DPGS for design parameter 1, see Fig. 1)

those primary design parameters that present the least possibility of creating additional violations.

It is assumed that the designer has full access to all primary design parameters whose values can liberally modify to obtain:

1. the calculation of the rest design parameters and
2. The relaxation of the occurring constraint violations.

If this access is constrained, then the method cannot operate because the constrained primary parameter(s) introduce some degree of automation in the design process.

For the proposed method, all possible relations between the violated parameter and the primary related parameters are taken into account. The method combines four individual criteria to produce a sorted list of primary design parameters. The designer can liberally choose any of the primary design parameters contained in this list to change its value. However, by choosing the first one, the possibility for creating additional violations to other design parameters is kept to a minimum.

The proposed method can operate either during the creation of the design solutions or after that. Because the mechanism used for the creation of design solutions is a simple depth-first method, every visited parameter can be checked for violation of its value in parallel with its value assignment. The search method is exhaustive, so it is ensured that all the necessary parameters will be visited and checked for violations. This implies that after a value assignment, a sorted violation list will be created given that at least one constraint violation occurs.

Sometimes, when one or more design solutions have been already produced, it becomes necessary to modify the values of one or more primary parameters in order to get solution alternatives. This modification forces recalculation of the nonprimary design parameters. For this case, any violation arising from the recalculation will be immediately reported as a sorted list of primary parameters.

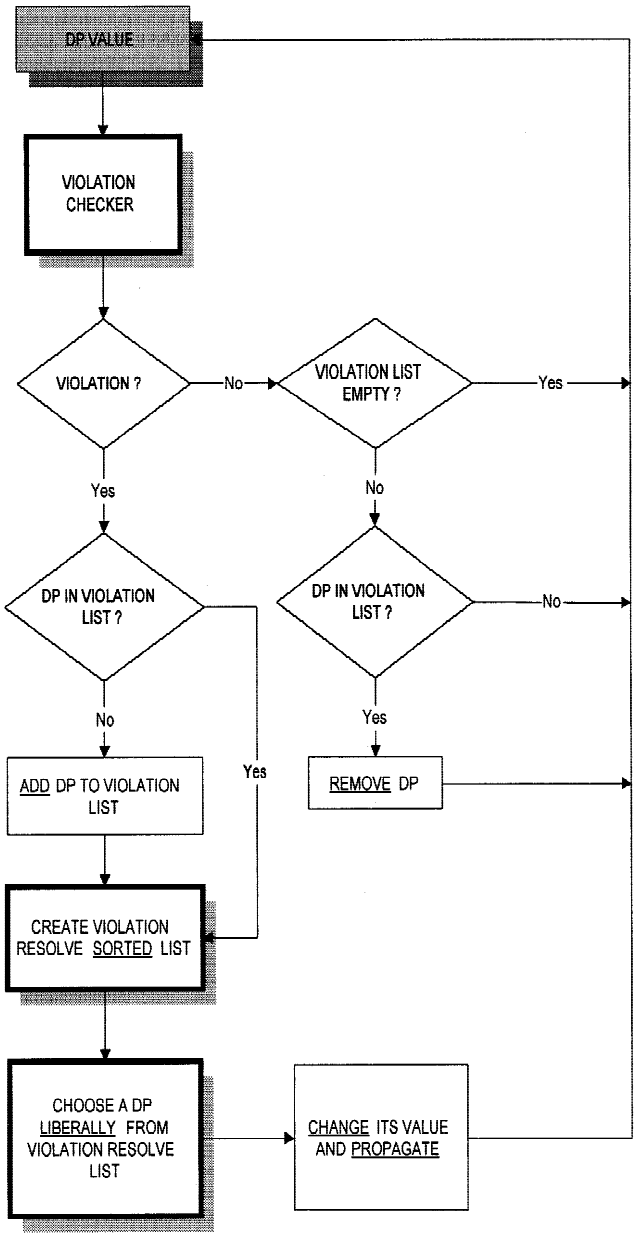

Fig. 4. The procedure for creating sorted primary parameters' list.

The program used for testing the method was created in Microsoft Visual Basic version 5 environment. It was developed as a module of an already existing design environment EXTDSEARCH used for producing solutions in well-defined design problems.

Although not implemented in the present paper, the method may easily apply to a more abstract level of problem decomposition. If a design main task is divided into subtasks and each subtask into more subtasks etc., then mul-

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press

tiple design task graph structures may be created. Because a task is considered to be violated if at least one of its design parameters is violated, the method may create a sorted list of subtasks that are related to and affect the violated one. Then the designer knows into which subtask to intervene first to resolve the violation without producing more violated subtasks. This approach is currently under investigation.

REFERENCES

Banares-Alcantara, R. (1991). Representing the engineering design process: Two hypotheses. *Computer Aided Design 23(9)*, 595–603.

Bitner, J., & Reingold, E. (1975). Backtracking programming techniques. *Communications of the ACM !8*, 651–655.

Bose, D.K., & Ellacot, S.W. (1991). Mathematical foundations of artificial intelligence. In *Artificial Intelligence in Engineering*, (Winstanley, G., Ed.), pp. 356–363. John Wiley & Sons Ltd., New York.

Brown, D.C. (1985). Failure handling in a design expert system. *Computer Aided Design 17(9)*, 436–442.

Brown, D.C., & Chandrasekaran, B. (1992). Investigating routine design problem solving. In *Artificial Intelligence in Engineering Design*, Tong, C., & Sriram, D., Eds.), Vol. I, pp. 221–249. Academic Press, Inc., San Diego, California.

Cross, N. (1991). *Engineering design methods*. J. Wiley & Sons, Chichester.

Dentsoras A.J. (1996). An approach of routine design based on extensive design space search. *Proc. Conf. EXPERSYS-96*, IITT-Inter., 115–120.

Dym, C.L. (1994). *Engineering design: A synthesis of views*. Cambridge University Press, New York.

Freuder, E., & Quinn, M. (1985). Taking advantage of stable sets of variables in constraint satisfaction problems. *Proc. of the 9th Int. Conf. Artificial Intell.*, 1076–1078.

Kumar, V. (1992). Algorithms for constraint satisfaction problems: A survey. *AI Magazine 13(1)*, 32–44.

Marcus S., Stout, J., & McDermott, J. (1988). VT: An expert elevator designer that uses knowledge-based backtracking. *AI Magazine 9(1)*, 95–112.

Mittal, S., & Araya, A. (1992). A knowledge-based framework for design. In *Artificial Intelligence in Engineering Design*, (Tong, C., & Sriram, D., Eds.), Vol. I, pp. 273–293. Academic Press, Inc., San Diego, California.

Nadel, B.A., & Lin, J. (1992). Automobile transmission design as a constraint satisfaction problem: First results. In *Artificial Intelligence in Engineering Design*, (Tong, C., & Sriram, D., Eds.), Vol. I, pp. 117–134. Academic Press, Inc., San Diego, California.

Rich, E., & Knight, K. (1991). *Artificial intelligence*. McGraw-Hill, New York.

Serrano, D., & Gossard, D. (1988). Constraint management in MCAE. In *Artificial Intelligence in Engineering: Design*, (Gero, J.S., Ed.), pp. 217–240. Elsevier, New York.

Serrano, D., & Gossard, D. (1992). Tools and techniques for conceptual design. In *Artificial Intelligence in Engineering Design*, (Tong, C., & Sriram, D., Eds.), Vol. I, pp. 71–116. Academic Press, Inc., San Diego, California.

Taylor, N.K., & Corlett, E.N. (1993). An expert system which constrains design. *Artificial Intelligence in Engineering Design 12*, 73–76.

Tong, C., & Sriram, D. (1992). Introduction. In *Artificial Intelligence in Engineering Design*, (Tong, C., & Sriram, D., Eds.), Vol. I, pp. 1–70. Academic Press, Inc., San Diego, California.

Tsalides, S., & Dentsoras, A.J. (1997). Application of design parameters space search for belt conveyor design. *J. Eng. App. Artif. Intell. 10(6)*, 617–629.

Ullman, D.G. (1992). *The mechanical design process*. McGraw-Hill, New York.

Argiris J. Dentsoras received his Ph.D. from the Mechanical Engineering Department of the University of Patras in Greece. Since 1993, he has been an Assistant Professor in the Dept. of Mech. Eng. and Aeronautics of the above university. His main interest over the last years has been the application of Artificial Intelligence Techniques and Methods in the domain of Mechanical Engineering and especially in the field of Machine Design. Together with other contributors, he developed some new methodologies on Routine Design and a relevant integrated environment that was used for designing belt conveyors. He participates in various joint research programs.

https://doi.org/10.1017/S0890060499133067 Published online by Cambridge University Press