



**HAL**  
open science

## Faster Generation of Feasible Design Points

Bernard Yannou, Faysal Moreno, Henri Thevenot, Timothy Simpson

► **To cite this version:**

Bernard Yannou, Faysal Moreno, Henri Thevenot, Timothy Simpson. Faster Generation of Feasible Design Points. ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Sep 2005, Long Beach, France. pp.355-363, 10.1115/DETC2005-85449 . hal-04506462

**HAL Id: hal-04506462**

**<https://hal.science/hal-04506462>**

Submitted on 15 Mar 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

## FASTER GENERATION OF FEASIBLE DESIGN POINTS

Bernard Yannou\*  
Faysal Moreno

Ecole Centrale Paris  
Laboratoire Génie Industriel  
Grande Voie des Vignes  
92295, Châtenay-Malabry, France  
[bernard.yannou@ecp.fr](mailto:bernard.yannou@ecp.fr); [faysal.moreno@lqi.ecp.fr](mailto:faysal.moreno@lqi.ecp.fr)

Henri J. Thevenot  
Timothy W. Simpson

Industrial Manufacturing and  
Mechanical & Nuclear Engineering  
The Pennsylvania State University  
University Park, PA 16802 USA  
[henri@psu.edu](mailto:henri@psu.edu); [tws8@psu.edu](mailto:tws8@psu.edu)

### ABSTRACT

Design space exploration during conceptual design is an active research field. Most approaches generate a number of feasible design points (complying with the constraints) and apply graphical post-processing to visualize correlations between variables, the Pareto frontier or a preference structure among the design solutions. The generation of feasible design points is often a statistical (Monte Carlo) generation of potential candidates sampled within initial variable domains, followed by a verification of constraint satisfaction, which may become inefficient if the design problem is highly constrained since a majority of candidates that are generated do not belong to the (small) feasible solution space. In this paper, we propose to perform a preliminary analysis with Constraint Programming techniques that are based on interval arithmetic to dramatically prune the solution space before using statistical (Monte Carlo) methods to generate candidates in the design space. This method requires that the constraints are expressed in an analytical form. A case study involving truss design under uncertainty is presented to demonstrate that the computation time for generating a given number of feasible design points is greatly improved using the proposed method. The integration of both techniques provides a flexible mechanism to take successive design refinements into account within a dynamic process of design under uncertainty.

**Keywords:** constraint programming, Monte Carlo simulation, interval analysis, uncertainty, design space exploration.

### INTRODUCTION

A *design concept* may be considered or defined (in a somewhat restrictive manner) as a parameterized model that links a set of *design parameters* or *variables* (referred to as DVs) to a set of *performance variables* (referred to as PVs). Consequently, a design concept does not refer to a unique design solution but to a set of potential design solutions. Being able to characterize the ability of a design concept to meet a set

of requirements is referred to as *set-based design* [1-3]. Set-based design opposes *point-to-point design*, i.e., the traditional trial-and-error process of dimensioning a design concept. Set-based design facilitates concurrent engineering and increases the quality of a design while lowering the number of design iterations. One can also refer to *design under uncertainty* where the challenge is to be able to look ahead at the properties of the subset of design points depending on a given design concept to ensure that a design satisfies the constraints and performance requirements. This process of assessing the potential of a design concept to meet the functional requirements frequently consists of an exploration of the solution space, i.e., the space of *feasible design points* pertaining to solutions that satisfy the current constraints.

Figure 1 summarizes the issue of concept dimensioning. Starting with functional requirements, FRs, a design concept is synthesized or proposed. This concept is structurally defined by a number of design variables (DVs) that may be relatively constrained (e.g., geometrically). For a given set of DV values, current values of performance variables (PVs) can be calculated (in case of explicit models of performances) or estimated (through expertise or real and/or virtual metamodeling, see for example Ref. [4]). In the traditional trial-and-error design process, current performance values are compared to expected performance values, called functional requirements (FRs). This comparison first checks that performance values fit within allowable bounds, which is the process of determining if the studied design point is a *feasible* one and if it lies within the solution space. A second outcome of this comparison is to determine if the studied design point is an optimal Pareto solution and/or to estimate its overall utility through a user-specified preference aggregation model.

---

\* Please address all correspondences to this author.

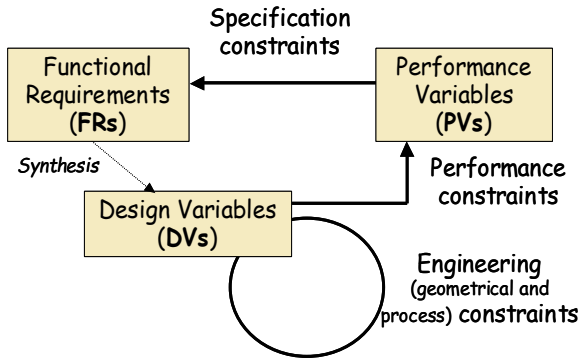


Figure 1. The dimensioning process for a design concept

Very simple tools can be used to explore the variability of the dimensioning of a given concept. Simpson has proposed a number of graphical user interfaces for dimensioning design concepts. For instance, an I-beam dimensioning example from Ref. [5] is shown in Figure 2. The graphical user interface allows designers to propose and manipulate values for the DVs using slider bars, and the corresponding performance parameters are plotted in the 2D window. The design solutions represented in the performance space can be flagged a posteriori as feasible or infeasible, depending on the current DV values that satisfy the allowable bounds or the constraints on the I-beam's design (see Figure 2); see Ref. [5] for more details.

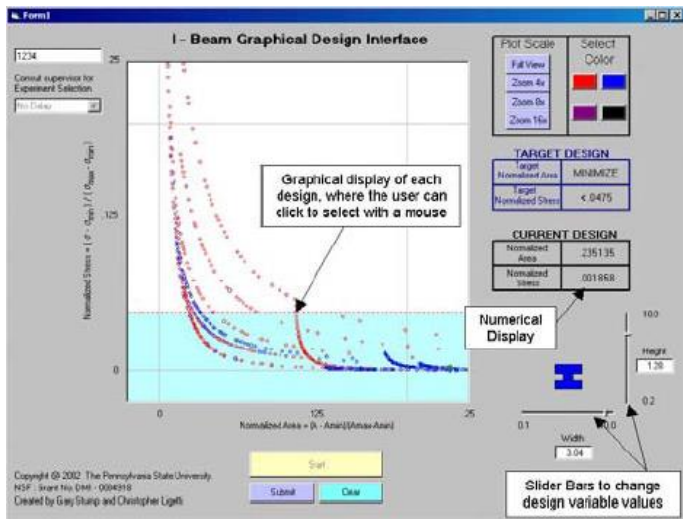


Figure 2. Interface for dimensioning an I-beam [5]

More evolved design space exploration tools exist, such as the ARL Trade Space Visualizer (ATSV), see Ref. [6]. After generating a set of design solutions using either Monte Carlo simulation or more rigorous Design of Experiments (DoE) techniques, the solutions are automatically displayed within the ATSV (see Figure 3). The selection of feasible design solutions is automated since solutions that do not *a posteriori* respect all of the constraints can be eliminated from the solution space for further exploration. Numerous graphical post-processing tools

are available in the ATSV to better understand the solution space: abstracting more than 3 dimensions into a 3D representation using colors, point shapes, textures to emulate extra dimensions, etc. (see Figure 3), finding Pareto optimal solutions, and representing the preference of design solutions.

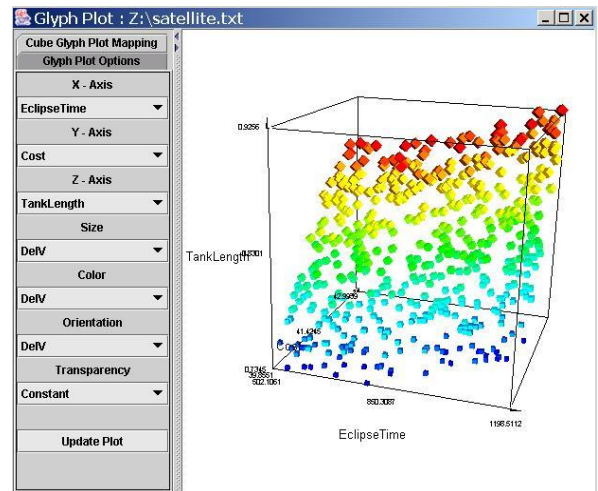


Figure 3. A graphical representation of the solution space in ARL Trade Space Visualizer [6]

Another interesting design space exploration mechanism exists in Ref. [7] which allows designers to explore a set of statistical runs by capturing the correlations between the design (parameters & performance)-value-tuples through sub-windows of interest on domains (see Figure 4). Both types of solution space explorers [6,7] require a minimal number of feasible design points and benefit from homogeneous point densities. The more constrained the design concept, however, the smaller the solution space will be and the less likely one is to generate a good set of feasible design points from the initial DV ranges.

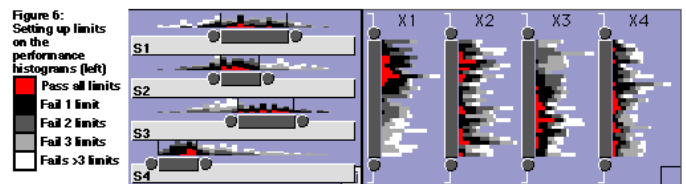
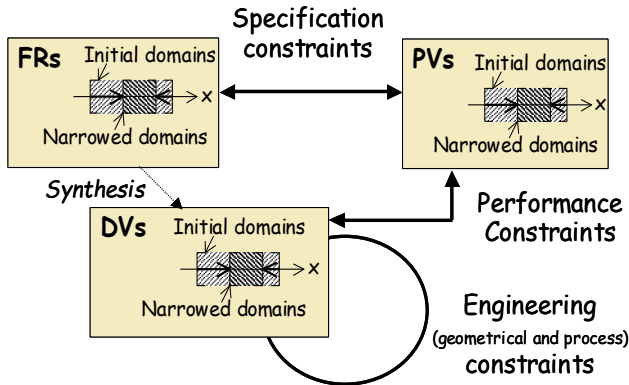


Figure 4. Performance (left) and parameter (right) histograms [7]

In the next section, the principles of Constraint Programming (CP) computation and of its coupling with design point generation are presented, provided that the constraints are expressed in an analytical form. Then, a case study involving dimensioning of a truss is given: the design constraints are provided and different stages of CP computation are explained. In next section, different strategies for generating feasible design points are considered depending on the use of a more or less sophisticated result from the CP computation. Finally, the conclusions highlight how this integration is well suited to a refinement process when designing under uncertainty.

## USING A PRIMARY CONSTRAINT PROGRAMMING COMPUTATION

With *Constraint Programming (CP) over reals*, uncertain *performance* and *design variables* are modeled as intervals of allowable values. These *constrained variables* may be equated to uniform distributions of values. CP techniques consist of sophisticated evolutions of interval analysis or *interval arithmetics* (see Ref. [8]) applied on a set of analytical constraints. Starting from a set of *initial domains* for the constrained variables and from a set of mathematical constraints linking the variables, different CP *consistency* or *filtering techniques* (such as *Hull*, *Box*, *weak-3B* or *3B*, see Refs. [9-11]) try to contract as much as their consistency degree allows the variable domains so as to eliminate infeasible values. This domain contraction stage is called the *filtering* stage. One tries to result in the most tightened Cartesian product of intervals, ensuring at any moment that any feasible solution is kept inside. This last important property refers to the *completeness* property and guarantees that the contraction process results in an outer design space approximation. This phenomenon is illustrated in Figure 5 through the two-way propagation of uncertainty reduction: from DVs to PVs (*analysis* direction) and from PVs to DVs (*synthesis* direction).



**Figure 5. The dimensioning issue for a design concept from a Constraint Programming perspective**

In the second stage, the mechanism of *domain splitting* (bisection for instance) is recursively applied in parallel with the *filtering* mechanism. A *search tree* is built until a stopping criterion (e.g., width of the domains, number of solutions) is reached. This *branch-and-prune* algorithm allows pruning out large parts of the design space whenever a domain is found to be empty. At the end of the process, the design space is approximated by a number of elementary Cartesian products of small intervals, denoted as *boxes*. The resulting *hull of boxes* provides the designer with valuable information about the potential values remaining for any design variable at this stage.

Finally, a graphical representation of this *collection of n-dimensional boxes* ( $n$  being the number of constrained design variables) is easy and convenient for obtaining good pictures of the resulting design space. The design space can be represented

by its two or three-dimensional projections on pairs or triplets of design variables (see Ref. [12]).

Table 1 illustrates the four outer approximations of the design space that we further consider when generating feasible design points, namely,

- The *initial domains*;
- The *filtered domains* after the uncertainty reduction propagation has been made for the first time;
- The *hull of boxes*, i.e., the projection on variable domains of the collection of boxes that have not been considered inconsistent after the domain splitting process (with no guarantee of any actual solution inside); and
- The *collection of boxes* itself.

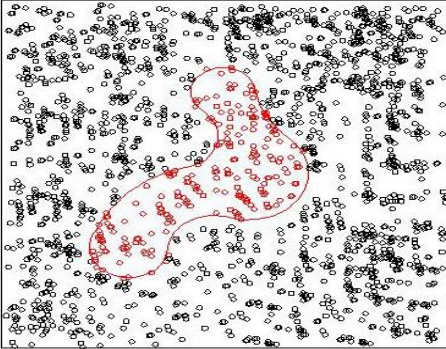
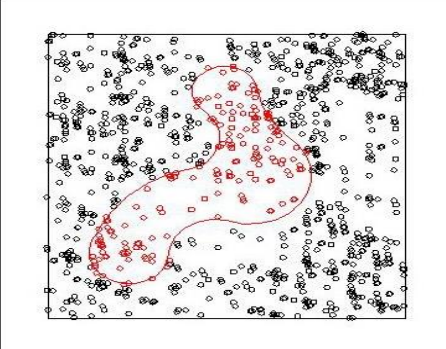
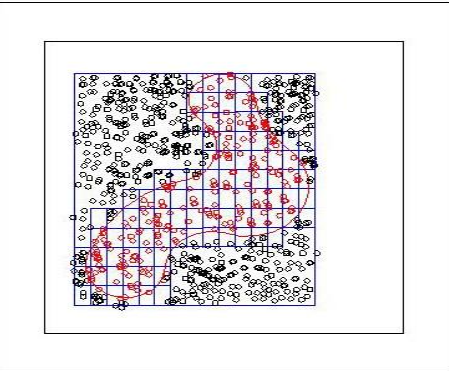
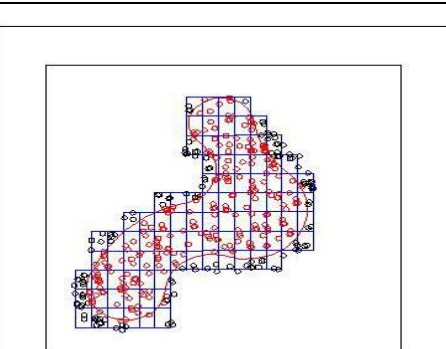
It is obvious that, in considering the outer approximations in that order, the finer the approximation, the faster a subsequent generation of feasible design points (represented in red inside the curved-bean-shaped design space). This relates to the efficiency of the global process, which is discussed later.

A detailed description of CP consistency techniques, of the branch-and-prune algorithm, and of its tuning is beyond the scope of this paper. We refer the reader to a previous paper that contains these details (see Ref. [13]). Let us simply mention that in this paper:

- We have used the CP platform RealPaver (see Ref. [10]), developed by the IRIN Computer Science Department of the Nantes (France) University.
- CP computations are performed using the *weak-3B consistency* technique that has been proven in Ref. [13] to be an efficient and convenient technique for mechanical design problems.
- The 3D representations of design spaces have been made with the tool *Universal Solution Viewer (USV)* [14].



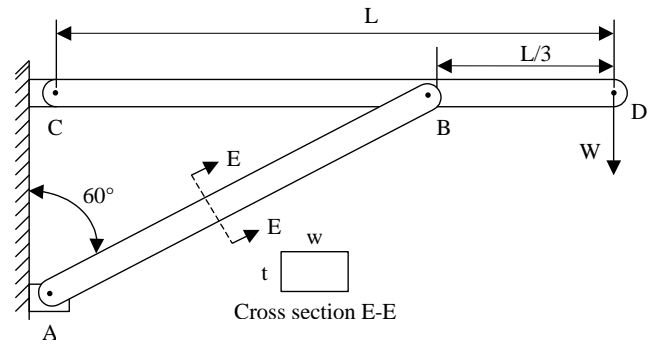
**Table 1. The four outer approximations of the design space (red curved bean) considered in this study**

Initial domain	
Filtered domains	
Hull of boxes (after domain splitting )	
Collection of boxes	

## A TRUSS DIMENSIONING DESIGN EXAMPLE

### Definition of the problem

Our case study consists of dimensioning the two members of the truss structure shown in Figure 6. This problem was originally proposed by Wood, et al. [15] to compute imprecise performance parameters from imprecise design parameters via fuzzy techniques. This example has also been used by Scott, et al. [16] in a different parameterized form to select an optimal Pareto solution that could not be selected via a linear aggregation function using importance weights. For this example, we use the exact parameterization and initial design variable ranges of the truss structure described by Wood, et al. [15], but we have chosen the more complex design constraints and performance parameters used by Scott, et al. [16].



**Figure 6. The parameterization of the truss structure**

The requirement is to design a mechanical structure supporting an overhanging vertical load at a distance  $L$  from the wall with a minimal mass. One possible configuration (see Figure 6) consists in a two-member pin-jointed bracket with a horizontal member ( $CD$ ) and a compression member ( $AB$ ) attached to the wall at an angle of sixty degrees. The common pin is located at two thirds of  $L$  from the wall. Both members have rectangular cross sections:  $w_{AB} \times t$  for ( $AB$ ) and  $w_{CD} \times t$  for ( $CD$ ),  $w$  standing for width and  $t$  for thickness. Additional design decisions have been made: the material of both members is steel, and we impose  $w_{CD} = w_{AB} - 0.025$ . The designer has to make decisions for the values of the following *design variables*:  $t$ ,  $w_{AB}$  and  $L$ . Moreover, the specification of the overhanging load  $W$  is imprecise, varying from 15-20 kN; consequently,  $W$  is treated as a fourth design variable.

The two *mechanical constraints* to satisfy are:

- the maximum bending stress,  $\sigma_b$ , in member ( $CD$ ) must be less than or equal to the allowable bending limit,  $\sigma_r$  (here 225 MPa for steel).
- the compression force  $F_{AB}$  in member ( $AB$ ) must be less than or equal to the buckling limit  $F_b$ .

The maximum bending stress,  $\sigma_b$ , is located at point B (see the bending moment diagram in Figure 7) and is given by the following formulas involving  $W_{CD}$ , the weight of member ( $CD$ ):

$$\sigma_b = \frac{2L \left( W + \frac{W_{CD}}{6} \right)}{w_{CD} t^2} \quad \text{with} \quad \begin{cases} W_{CD} = \rho g w_{CD} t L \\ w_{CD} = W_{AB} - 0.025 \end{cases} \quad (1)$$

The compression force in member (AB) is given by the following formulas involving  $W_{AB}$ , the weight of member (AB):

$$F_{AB} = \sqrt{\left\{ \frac{9}{2\sqrt{3}} \left( W + \frac{W_{CD}}{2} + \frac{W_{AB}}{3} \right) \right\}^2 + \left\{ \frac{3}{2} \left( W + \frac{W_{CD}}{2} \right) \right\}^2} \quad (2)$$

$$\text{with} \quad W_{AB} = \rho g w_{AB} t L_{AB} \quad \text{and} \quad L_{AB} = \frac{4\sqrt{3}}{9} L$$

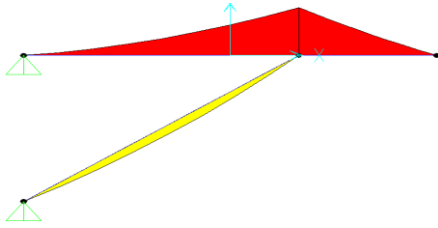


Figure 7. The bending moment in the truss

The buckling limit in member (AB) is given as:

$$F_b = \frac{\pi^2 E I_{AB}}{L_{AB}^2} = \frac{9\pi^2 E w_{AB} t^3}{64 L^2} \quad (3)$$

The *performance variables* are the mass  $M$  of the structure (to be minimized) and the safety factor,  $s$ , i.e., the amount of over-dimensioning beyond the satisfaction of the two mechanical constraints. The mass  $M$  is given by:

$$M = W_{AB} + W_{CD} \quad (4)$$

The safety factor of the truss structure  $s$  is the minimum between the safety factor below the allowable bending limit,  $\sigma_b$ , namely,  $s_\sigma$ , and the safety factor below the buckling limit,  $F_b$ , namely,  $s_F$ , which is expressed as:

$$s_\sigma = \frac{\sigma_r}{\sigma_b}, \quad s_F = \frac{F_b}{F_{AB}}, \quad s = \min(s_\sigma, s_F) \quad (5)$$

The two mechanical constraints may be merely expressed by:  $s_\sigma \geq 1$ ,  $s_F \geq 1$  or simply by the single constraint:

$$s \geq 1 \quad (6)$$

### Modeling the Constraint Programming problem

To define a Constraint Programming problem, the initial domains (ranges) of the *performance* and *design variables* must be defined; Table 2 summarizes these for this example. The design variables domains are those defined by Wood, et al. [15]. In fixing the lower bound of the safety factors  $s_\sigma$ ,  $s_F$  and  $s$  to 1, the mechanical constraints are taken into account. The lower bound of mass is simply set to 0 with no further information.

Table 2. Initial ranges of design and performance variables

Design variables	Performance variables	Constants
$t \in [0.04, 0.10]$	$M \in [0, +\infty[$	$E = 207 \cdot 10^9 \text{ Pa}$
$w_{AB} \in [0.04, 0.13]$	$s_\sigma \in [1, +\infty[$	$\rho = 7830 \text{ kg/m}^3$
$L \in [3, 4]$	$s_F \in [1, +\infty[$	$g = 9.81 \text{ m/s}^2$
$W \in [15000, 20000]$	$s \in [1, +\infty[$	$\sigma_r = 225 \cdot 10^6 \text{ Pa}$

An important rule in the modeling of a CP problem is that intermediary variables that designers are not interested in must be eliminated from the set of constraints so as not to consider those variables in the design space and to get the best domain contraction for the actual design and performance variables. This is why:

- Variables  $W_{AB}$ ,  $W_{CD}$ ,  $w_{CD}$ ,  $F_b$  and  $L_{AB}$  must be replaced by their expressions in  $t$ ,  $w_{AB}$ ,  $L$  and  $W$  in the constraints.
- Variables  $s_\sigma$  and  $s_F$  must not be considered as performance variables in which the designer is interested, and they must not be bisected during the splitting process. A special mechanism exists in RealPaver [10] for hiding such variables in the enumerated boxes. When these variables are defined as functions of other variables, it symbolically replaces internal occurrences of the variables by their expressions, even though the initial domains may be defined on them. The intermediary variables to be hidden are preceded by a \$ sign in Eq. (7).
- In addition, the number of occurrences of the same variable must be minimized as much as possible to avoid the *dependency problem* (see Ref. [11]). The *dependency problem* occurs due to the fact that a variable occurrence is suddenly replaced by its current domain during the solution process. Subsequently, the multiple occurrences of a given variable within a given constraint and even between different constraints are decorrelated. This decorrelation results in relaxed constraints and then in larger domains. This is why it is often necessary to reformulate constraints by decreasing the number of occurrences of the same variable by appropriate factorization strategies (see Ref. [17]). The choice of the *weak-3B-consistency* global consistency technique (as opposed to *local consistency techniques* like *hull* or *box*) partly overcomes this problem (see Refs. [13] and [18]).

Finally, the constrained problem is entirely expressed by the four following constraints:

$$\begin{cases}
 \$s_\sigma = \frac{\sigma_r (w_{AB} - 0.025)^2}{2L \left( W + \frac{1}{6} \rho g t L (w_{AB} - 0.025) \right)} \\
 \$s_f = \frac{\left( \frac{3\pi^2 E w_{AB}^3}{32 L^2} \right)}{\sqrt{3 \left( W + \frac{1}{2} \rho g t L \left( w_{AB} \left( 1 + \frac{8}{9\sqrt{3}} \right) - 0.025 \right) \right)^2 + \left( W + \frac{1}{2} \rho g t L (w_{AB} - 0.025) \right)^2}} \\
 M = \rho g t L \left( w_{AB} \left( 1 + \frac{4\sqrt{3}}{9} \right) - 0.025 \right) \\
 s = \min(\$s_\sigma, \$s_f)
 \end{cases} \quad (7)$$

### The design scenario

After considering the sole constraint  $s \geq 1$ , and starting from initial domains of Table 2, the first filtering stage followed by the domain splitting stage into 1000 boxes has led to the first collection of boxes and hull of boxes visible in the first row of Table 3 (Case #1). This first CP computation already leads to noticeable domain reduction.  $M$  is ensured to stand between 2077.9 and 6300.9 N, the safety factor is already guaranteed to be lower than 2.567, and the lower bounds of  $t$  and  $w_{AB}$  have been tightened. The shape of the design space (using 3D box projections of  $\{t, w_{AB}, L\}$ ,  $\{L, W, M\}$ , and  $\{W, M, s\}$ ) confirms some intuitive trends: the greater the supported weight  $W$  or the structure length  $L$ , the greater the structure mass  $M$  and the lower the safety factor. However, the fact that there seems to exist a given safety factor for which the mass  $M$  is the greatest is not so intuitive and reveals that simultaneously minimizing  $M$  and maximizing  $s$  is a difficult task (see Ref. [16]).

If the designer notes that there is a sufficient degree of freedom for further constraint on the design, then the specifications on  $M$  and  $s$  can be strengthened. Increasing the lower bound of the safety factor to 1.5, given that the safety factor is close to 1, is risky; therefore, let us impose  $M \leq 3200$  on the structure mass to control it better. After recomputation, the design space has been dramatically reduced. As a first consequence, the length  $L$  can no longer be 4 m; it is limited to 3.15 m. The safety factor domain is importantly tightened, and  $s$  is guaranteed to be lower than 1.664. The smallest mass  $M$  is now 2926.5 N. The domains of  $w_{AB}$  and  $W$  are also tightened considerably. The designer is now informed that the overhanging load can no longer be greater than 16638 N. It has been useful to keep variability on the specified load since this flexibility has been used to find a better design.

### IMPROVED STRATEGIES FOR GENERATING FEASIBLE DESIGN POINTS

We would like to globally measure the efficiency of a CP computation of an outer approximate design space in the generation of a given number of feasible design spaces in two cases:

- The case of a “not so constrained” design problem, which means that the initial domain is not large compared to the effective solution space. This is the situation of **Case #1** of the specification constraints on the truss structure.
- The case of a “highly constrained” design problem, which means that the initial domains are much larger than the actual solution space. This is the case of **Case #2** of the specification constraints on the truss structure.

Four strategies are considered for generating design points when assuming uniform distributions on design variables  $\{t, w_{AB}, L, W\}$ :

- *From initial domains* (see Table 2). This amounts to a priori totally ignoring the location of feasible design points.
- *From the hull of boxes of Case #1* (see Table 3). This hull of boxes is a first outer approximation of the design space corresponding to the constrained system:  $s \geq 1$ .
- *From the hull of boxes of Case #2* (see Table 3). This hull of boxes is a first outer approximation of the design space corresponding to the highly constrained system:  $s \geq 1.5, M \leq 3200$ .
- *From the collection of boxes of Case #2* (see Table 3). This corresponds to a second finer outer approximation of the constrained system:  $s \geq 1.5, M \leq 3200$ .

This last generation strategy was required to implement a specific algorithm. The collection of boxes issued from the CP computation will be disjointed by construction (see Figure 8 for a textual representation). The first operation consists of projecting this collection of boxes over the subspace of design variables  $\{t, w_{AB}, L, W\}$ . Next, for all of the *remaining boxes*, the volume is computed. In considering the total volume and the approximate total number of generated design points (here 100,000 trials are expected), an approximate number of trials is calculated for each remaining box in proportion of its volume. Finally, this calculated number of trials is sampled within the considered remaining box to ensure a constant density of generated design points in the subspace of design variables  $\{t, w_{AB}, L, W\}$ .

**Table 3. CP computation of the truss structure - considering two series of specification constraints**

Specification Constraints	$\{t, w_{AB}, L\}$ projection	$\{L, W, M\}$ projection	$\{W, M, s\}$ projection	Hull of boxes
Case #1 $s \geq 1$				$t \in [0.0621, 0.1]$ $w_{AB} \in [0.0654, 0.13]$ $L \in [3, 4]$ $W \in [15000, 20000]$ $M \in [2077.9, 6300.9]$ $s \in [1, 2.567]$
Case #2 $s \geq 1.5$ $M \leq 3200$				$t \in [0.0887, 0.1]$ $w_{AB} \in [0.0859, 0.1026]$ $L \in [3, 3.150]$ $W \in [15000, 16638]$ $M \in [2926.5, 3200]$ $s \in [1.5, 1.664]$

```

OUTER BOX 1
L in [3 , 3.039020413403462]
wab in [0.08592855543083494 , 0.08709219131357029]
t in [0.09905743560047796 , 0.1]
W in [15028.1951508509 , 15042.29272627635]
M in [2926.539056545716 , 2947.778741474205]
s in [1.5 , 1.527422371332263]

precision: 21.2, elapsed time: 59,440 ms

OUTER BOX 2
L in [3 , 3.039068372366211]
wab in [0.08592855543083494 , 0.08867264526592694]
t in [0.0978295013131977 , 0.1]
W in [15000 , 15014.09757542545]
M in [2947.778741474205 , 2974.328347634815]
s in [1.5 , 1.555974882162037]

precision: 26.5, elapsed time: 59,440 ms

OUTER BOX 3
L in [3 , 3.039068372366211]
wab in [0.08592855543083494 , 0.08867264526592694]
t in [0.0978295013131977 , 0.1]
W in [15014.09757542545 , 15028.1951508509]
M in [2947.778741474205 , 2974.328347634815]
s in [1.5 , 1.555974882162037]

precision: 26.5, elapsed time: 59,440 ms

```

**Figure 8. Textual representation of a collection of boxes**

Table 4 provides the ratios of design points that have been detected as feasible, i.e., respecting the specification constraints in Case #1 and Case #2. The results confirm our expectations.

For the “not so constrained” design problem (Case #1), the initial ratio of feasible design points to total number of points generated is satisfactory with 15% because the design space is still large relatively to the initial domains, and many design points directly fall within the design space. When considering a better outer approximation of the design space with the hull of boxes of Case #1, this ratio is significantly increased to 33%.

For the “highly constrained” design problem (Case #2), the ratio of feasible design points is dramatically low since only 4 trials over 100,000 were respecting the specifications of Case #2 when starting from the initial domains. Indeed, the design space is much narrower, compared to the initial domains, due to strong constraints (see Table 3). This ratio reaches 1.7% when the design points are generated from the hull of boxes of Case #2. This low score compared to Case #1 (33%) means that many Cartesian products within the hull domains are not valid, and consequently that the design space is less complicated. Finally, this latter ratio is improved by a factor of 5 to reach 9.5% when the design points are generated from the collection of boxes finely approximating the design space. This final score is satisfactory since Case #2 represents a difficult case. Theoretically, this last ratio could reach 100% provided that:

- The filtering technique is highly consistent (a technique is highly consistent at the condition that any box containing no design point is detected inconsistent and ruled out from the list of solution boxes); and
- The domain splitting is infinitely fine.

For example, this ratio of 9.5% would be improved by adopting a 3B-consistency filtering technique instead of a weak-3B-consistency technique (see Ref. [13] for a comparison of techniques’ efficiency) and in enumerating 2000 boxes instead of 1000 boxes. However, the computation time for CP would increase needlessly because of a posteriori constraint checking from worse design point generation would be more time-saving.

The computation time for generating the feasible design points mentioned in table 1 is the sum of:



- time  $t_1$  for the constraint programming computation which varies here from 49 to 70 seconds (not applicable to column #1 of table 1),
- time  $t_2$  for the Monte-Carlo generation of 100,000 design points within allowable interval bounds and for the later corresponding constraint checking. Here, we have considered in a first approximation that an elementary constraint checking was constant in time, independently of the feasibility/infeasibility of the design point, leading to an approximate 3600 seconds in all cases (for 100,000 trials).

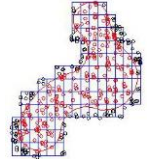
The extra time for the MC generation of design points from the collection of boxes turned out to be negligible in

comparison with the later constraint checking. Finally, the efficiency of the different cases must be compared for a given number of feasible design points found and an efficiency indicator may be defined by:

$$Efficiency = \frac{\text{Number of feasible design points}}{t_1 + t_2} \quad (8)$$

For the “not so constrained” design problem (Case #1), our method speeds up the generation of feasible design points by a factor of 2. But, For the “highly constrained” design problem (Case #2), our method speeds up the generation of feasible design points by a factor of 2300.

**Table 4. Ratios of feasible design points from 4 Monte Carlo generation strategies (over 100,000 trials)**

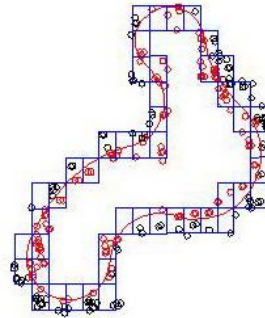
		<b>Initial domains</b> $t \in [0.04, 0.10]$ $w_{AB} \in [0.04, 0.13]$ $L \in [3, 4]$ $W \in [15000, 20000]$	<b>Hull of boxes Case#1</b> $t \in [0.0621, 0.1]$ $w_{AB} \in [0.0654, 0.13]$ $L \in [3, 4]$ $W \in [15000, 20000]$	<b>Hull of boxes Case#2</b> $t \in [0.0887, 0.1]$ $w_{AB} \in [0.0859, 0.1026]$ $L \in [3, 3.150]$ $W \in [15000, 16638]$	<b>Collection of boxes Case#2</b> 
<b>Case#1</b>	$s \geq 1$	<b>14.97%</b> (14,969 feasible) $t_2=3600s$ <b>4.16 feasible points / s</b>	<b>33.30%</b> (33,300 feasible) $t_1=49s, t_2=3600s$ <b>9.13 feasible points / s</b>		
<b>Case#2</b>	$s \geq 1.5$ $M \leq 3200$	<b>0.00%</b> (4 feasible) $t_2=3600s$ <b>0.0011 feasible points / s</b>	<b>0.01%</b> (12 feasible) $t_1=70s, t_2=3600s$ <b>0.0033 feasible points / s</b>	<b>1.70%</b> (1703 feasible) $t_1=59s, t_2=3600s$ <b>0.47 feasible points / s</b>	<b>9.47%</b> (9471 feas.) $t_1=59s, t_2=3600s$ <b>2.59 feasible points / s</b>

## CONCLUSIONS

The exploration of design concepts is a fundamental aspect of engineering design. Most of the time, it consists of sampling design points within initial domains of design variables. Next, these design points are detected as feasible whenever the constraints are checked, but in the case where the design problem is very constrained and where the designer has no idea of the size, shape, and/or location of the design space (in lieu of feasible design points), the ratio of the number of feasible design points generated to the total number of points generated can be close to 0. In the case where explicit constraints are available, Constraint Programming techniques may be used to proceed to a first computation of an outer approximation of the design space as a collection of disjointed boxes. We have shown that this first computation allows us to focus on the location of the design space for a targeted sampling of design points, which are much more likely to be feasible. A specific sampling algorithm has been developed from the result of a CP computation. Satisfactory results have been obtained for an example involving the design of a truss structure.

We believe that this mechanism of integrating a Constraint Programming computation and a Monte Carlo simulation of

design points is particularly advantageous in the case of successive strengthened constraints and successive zooms on the design space. This integration could also be a means to frame the frontier of the design space: externally by CP and internally by feasible design points (see Figure 9). Future work will consider larger problems to identify the trade-off between computation time and problem size that most likely exists.



**Figure 9. Framing of the frontier of design space**

## REFERENCES

- [1] Finch W.W., 1999, "Set-Based Models of Product Platform Design and Manufacturing Processes", *Proc. DETC'99: ASME 1999 Design Engineering Technical Conference*, Las Vegas, NV, USA, Paper No. DETC99/DTM-8763.
- [2] Finch W.W., Ward A.C., 1997, "A Set-Based System for Eliminating Infeasible Designs in Engineering Problems Dominated by Uncertainty", *Proc. DETC'97: ASME / Design Engineering Technical Conference*, Sacramento, CA, USA, Paper No. DETC97/DTM-3886.
- [3] Ward A.C., Liker J.K., Sobek D.K., Cristiano J.J., 1994, "Set-Based Concurrent Engineering and Toyota", *Proc. DETC'94: ASME / Design Engineering Technical Conference*, Sacramento, CA, USA, pp. 79-90.
- [4] Qian Z., Seepersad C.C., Joseph V.R., Jeff Wu, C. F., Allen J.K., 2004, "Building Surrogate Models based on Detailed and Approximate Simulations", *Proc. DETC/DAC: ASME Design Engineering Technical Conferences / Design Automation Conference*, Salt Lake City, UT, USA, Paper No. DETC2004/57486.
- [5] Barron K., Simpson T.W., Rothrock L., Frecker M., Barton R., Ligetti C., 2004, "Graphical User Interfaces for Engineering Design: Impact of Response Delay and Training on User Performance", *Proc. DETC/DTM: ASME Design Engineering Technical Conferences / Design Theories and Methodologies*, Salt Lake City, UT, USA, Paper No. DETC2004/DTM-57085.
- [6] Stump G., Yukish M.A., Simpson T.W., 2004, "The ARL Trade Space Visualizer: An Engineering Decision-Making Tool", *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA, AIAA-2004-4568.
- [7] Tweedie L., Spence R., Dawkes H., Su H., 1996, "Externalising Abstract Mathematical Models", *Proc. Conference on Human Factors in Computing Systems*, April 13-18, 1996, Vancouver, British Columbia, Canada.
- [8] Moore R.E., 1979, *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, PA, USA.
- [9] Benhamou F., Goulard F., Granvilliers L., Puget J.-F., 1999, "Revising Hull and Box Consistency", in *Proc. of ICLP'99*, The MIT Press, Las Cruces, NM, USA.
- [10] Granvilliers L., 2002, "RealPaver User's Manual (V0.2)", <http://www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver/main.html>, Nantes University, Lab of Computer Science IRIN, France.
- [11] Granvilliers L., Benhamou F., Huens E., 2001, "Constraint Propagation (Chapter 5)", in COCONUT Deliverable D1 - Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art, The Coconut Project, pp. 113-149.
- [12] Sam J., 1995, "Constraint Consistency Techniques for Continuous Domains", *Ph.D. Thesis* number 1423, Ecole Polytechnique Fédérale de Lausanne, EPFL, France.
- [13] Yannou B., Harmel G., 2004, "A Comparative Study of Constraint Programming Techniques over Intervals in Preliminary Design", *Proc. DETC/DAC: ASME Design Engineering Technical Conferences / Design Automation Conference*, Salt Lake City, UT, USA, Paper No. DETC2004/57152.
- [14] Christie M., 2002, "Universal Solution Viewer, USV: User Manual", IRIN Lab, Nantes University, France.
- [15] Wood K.L., Antonsson E.K., Beck J.L., 1989, "Computations with Imprecise Parameters in Engineering Design: Background and Theory", *ASME Journal of Mechanisms, Transmissions and Automation in Design*, **111**(4), pp. 616-625.
- [16] Scott M.J., Antonsson E.K., 2000, "Using Indifference Points in Engineering Decisions", *Proc. Of the ASME/DETC2000/DTM*, Baltimore, MD, USA, Paper No. DETC2000/DTM-14559.
- [17] Ceberio M., Granvilliers L., 2000, "Solving Nonlinear Systems by Constraint Inversion and Interval Arithmetic", *Proc. AISC'2000: 5th International Conference on Artificial Intelligence and Symbolic Computation*, Madrid, Spain, pp. 127-141.
- [18] Lhomme O., Gotlieb A., Rueher M., Taillibert P., 1996, "Boosting the Interval Narrowing Algorithm", *Proc. ICLP*, MIT Press, Cambridge, MA, USA.