Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 6861429, 14 pages
https://doi.org/10.1155/2018/6861429

Research Article

System Sizing with a Model-Based Approach: Application to the Optimization of a Power Transmission System

Pierre-Alain Yvars [1] and Laurent Zimmer [2]

[1] *Institut Supérieur de Mécanique de Paris (SupMéca), QUARTZ, 3 rue Fernand Hainaut, 93407 Saint-Ouen, France*
[2] *Dassault Aviation, Direction Générale Technique, 78 quai Marcel Dassault, 92552 Saint-Cloud, France*

Correspondence should be addressed to Pierre-Alain Yvars; pierre-alain.yvars@supmeca.fr

Received 20 March 2018; Accepted 21 June 2018; Published 26 July 2018

Academic Editor: Chaudry M. Khalique

We test the relevance of a model-based approach for sizing and optimizing complex systems. Classically a model-based approach is characterized by a clear partition between the problem description and the solving process. In the case of a design problem, we show that the sizing task could be systematically characterized and therefore could lead to a declarative model combining both system description and design requirements. Once translated into a constraint satisfaction problem, the resulting model can be solved with interval constraint programming methods and algorithms. Our first contribution to this approach is to precisely characterize the sizing task in design. The resulting terminology enables us to easily and systematically express the problem as a constraint satisfaction problem (CSP) which combines in the same model the system description and the design requirements. We have tested the approach on the optimal sizing problem of a power transmission system. Previous authors have described this scalable case study. They provide a mathematical formulation of the problem and solve it with an evolutionary algorithm. Starting from their description, we apply our methodology to model the problem as a CSP and then solve it with interval constraint programming algorithms. Our solutions are more adequate both in computational time and in optimization results than those published in the literature on the same problem. Moreover the declarative nature of constraint programming makes modifications or extensions easier than with evolutionary programming. The explanation of these results is our second contribution to the approach. However some important modelling issues remain to address in order to capture more and more complex system specifications. Further research is presented at the end of this paper.

1. Introduction

Following Pahl and Beitz [1], the design process can be divided into three main phases: conceptual design, embodiment design, and detailed design.

In this paper, we will focus on the embodiment design phase which is the first phase requiring calculus. More precisely, we intend to split the embodiment design phase into two subphases that will be called architectural generation and system sizing.

Broadly speaking, architectural generation fixes the structure of the system, i.e., the type and number of components used in the system building. System sizing focuses on setting the value of each physical and technological parameter.

Nowadays, virtually all products can be considered as systems. They are made up of components which are connected together and some components are themselves subsystems.

The resulting system has to fulfil the requirements deriving from the conceptual design phase such as physical performance. In the case of technical products, many other types of requirements matter like technological, normative, or even legal constraints.

The current paper lays emphasis on the use of computational methods for modelling technological system and for solving the system sizing problem.

This paper revolves around four chapters. The first offers to apply a model-based approach for system sizing. The second explains the scientific paradigm used to support our model-based system sizing method. This paradigm is known as constraint solving. The third presents the tool used to carry out this study and the last chapter is dedicated to the scalable case study of the sizing problem of a power transmission system.

2. System Sizing with a Model-Based Approach

In this chapter, we will point up a particular design phase which is commonly called "sizing". Putting it in informal terms, sizing a system is aimed at fixing its design in order to produce performances which will both fulfil the physical and technological constraints and meet the customer's requirements.

Over the last years, many authors have pointed out that the art of design can be considered as a kind of problem solving task [2] which relies heavily on all kinds of knowledge. In this perspective, the field has been widely investigated by Artificial Intelligence researchers. A lot of various reasoning techniques have been experimented on design applications such as rule-based, case-based, or model-based reasoning [3, 4].

Broadly speaking, rule-based systems better suit expert knowledge while case-based systems are more suitable to experimental knowledge and model-based reasoning suit better with knowledge of the physical world. In the following, we will narrow down the study to engineering systems. These systems are artefacts which follow the laws of the physical world; therefore using a model-based reasoning approach for designing them, particularly for sizing them, stands to reason.

A model-based reasoning approach puts emphasis on the development of an abstract model which represents the system. This model offers a basis for any kind of reasoning underlying the system; the reasoning task is processed by an "engine" dedicated to intended task (planning, diagnosis, sizing, etc.). This engine combines the model knowledge with additional data to derive the results required.

Many kinds of models may be used. Their choice depends both on the application domain and on the intended task. Previous works in various technical fields [5, 6] helped us to set some minimal characteristics of the models which are to be used for sizing and optimizing engineering systems.

Firstly these models must be quantitative because most parameters are continuous in engineering. Secondly, the appropriate behaviour of the targeted system has to be described. It may be (behaviour over time) dynamic behaviour or steady-state behaviour. Practically steady-state models are already of a great interest in design especially in preliminary design; hence we will limit our study to these hereafter. Thirdly we will set out some technological or regulation constraints applying to the system. Finally we will lay down the customer requirements that the system is required to fulfil. To summarize we need to build a quantitative model of the problem to solve (i.e., sizing or optimizing). This problem model is the combination of the steady-state model of the targeted system, the technological and regulation constraints model, and the requirements model.

2.1. Sizing Model Specification. Before giving further details about the structure of sizing models, let us now introduce some terminology that will be used further down. Many researchers in preliminary design [7, 8] represent the knowledge related to the product to be designed by using a set of typed parameters. It stands to reason that apart from types, parameters can be characterized by many other attributes such as dimension, unit, type of value, range of value, and so forth and some of them, especially dimension and unit, are very important design modelling characteristics but they are beyond the scope of the present paper. Focusing on the type, we observe that the number, name, and above all the semantic of the considered types depend on the authors. X. Fischer in his Ph.D. Thesis [9] makes a typical proposal based on 3 kinds of parameters: the "design parameters", the "physical parameters", and the "objective parameters". Their semantic is informally defined as follows. The "design parameters" are related to the structural description of the system and its conceptual definition. The "physical parameters" are related to the physical behaviour of the system. The "objective parameters" are performance indicators which enable one to validate the quality of the design. This proposal is completed by an additional classification of parameters in known parameters (i.e., with a value) and unknown parameters.

Table 1: Typology of variables.

Type\Value	Known	Unknown
design	Constant	Design Variable
physical	Constant	Behavioural Variable
objective	-	Performance Indicator

In a model-based approach, such a product description is destined to be translated into a mathematical model. We propose to refine the above model as follows by considering 4 types of elementary concepts:

(i) Unknown design parameters
(ii) Unknown physical parameters
(iii) Objective parameters
(iv) Known parameters

Known parameters are translated into any kind of constant; in an obvious way, unknown design parameters are translated into Design Variables and unknown physical parameters are translated into Behavioural Variables.

Objective parameters will not be translated into variables but in criteria functions of some design or Behavioural Variables which will be called performance indicators.

Finally Table 1 summarizes the new classification.

Let us give some examples in the field of mechanics. Assuming we have to design a mechanical product,

(i) geometrical dimensions or material characteristics will be expressed with DVs;
(ii) forces, movements, and energies will be expressed with BVs;
(iii) total weight and whole cost will be expressed with PIs.

Moreover, sizing models are not only made up of constants and variables but also of relations. Many relations of different nature connect both DVs and PVs through a set of

(i) equations expressing the laws of physics (e.g., conservation laws);

(ii) inequalities expressing technological limitations (e.g., elastic limit) or regulation constraints (e.g., “minimum of two independent power sources in the system”);

(iii) inequalities or optimization criteria expressing customer requirements (e.g., range min, mass minimization).

Finally a sizing model is a set of equations and inequalities linking together Design Variables and Behavioural Variables as it presents some significant characteristics:

(1) the set of equations is usually nonlinear;

(2) the number of inequalities can be greater than the number of equations;

(3) the model combines continuous and discrete variables.

Indeed, most of the equations which are part of the model express physics laws which are nonlinear; therefore the whole model itself is usually nonlinear. In engineering domains, the number of technological limitations or constraints of conformity to standards is ever increasing. In many models, some Design Variables related to technological parameters are discrete either intrinsically (e.g., the number of teeth of a gear wheel) or because of standards (e.g., normalized screw diameters).

2.2. Sizing Task Features. Following the previous definition of Design Variables and Behavioural Variables, sizing the system consists in finding an assignment of Design Variables and Behavioural Variables such that

(1) the assignment of Design Variables determines the structure of the system;

(2) the assignment of Behavioural Variables determines the behaviour of the system;

(3) the whole assignment is consistent with all the elements of the sizing Model.

In classical engineering fields, most system behaviours are deterministic and ruled by a mathematical system of “n” behavioural equations and “n” unknown variables. These variables correspond to the Behavioural Variables in our terminology. Therefore finding the behaviour consists in solving a square system of equations. A number of more or less sophisticated mathematical methods are dedicated to this solving problem which remains intrinsically difficult especially in the presence of nonlinear equations. The task addressing the rendering of behaviour is a “simulation” task. This terminology mainly refers to dynamic behaviour that will also use it hereafter for stationary behaviour.

Simulation can be and is effectively used in design, especially during the detailed design phase in order to check on the appropriate behaviour of the final product with respect to the requirements. However in earlier design phases (preliminary design, embodiment design) an important part of the problem is to determine the system structure of the future product itself. The task addressing this point is a sizing task. For a sizing task the number of behavioural equations remains the same but the number of unknown variables increases with the addition of some Design Variables to assign. Hence, we no longer obtain a square system but an underconstrained system of equations to solve. Therefore sizing a system is generally much more difficult than simulating it!

Since the system is underconstrained, many solutions to the problem can be found out. Therefore it is worthwhile to use optimization criteria in order to come up with the ultimate solution amongst all the solutions of the “design space”. It is important to notice that in such a search space solutions can radically differ from each other; therefore the optimization process has to be as global and exhaustive as possible in order not to be trapped in a local optimum.

2.3. Overview of Existing Methods and Tools. A range of numerical tools are used in engineering. Our purpose in this paragraph is to make a survey of some of the most popular software tools currently used by engineers for their preliminary design studies. For each tool or type of tool, we will investigate whether their mathematical solving methods can suit or not the sizing task.

2.3.1. Simulators. Simulators are more and more widely used in the study of systems. Many products with attractive user interfaces are hence available in various engineering domains (e.g., Simulink, Saber, AmeSim, and Modelica) and recent advances in modelling (multiphysics, higher level modelling language) aim at positioning them as favourite tools for system design and architecture evaluation like Modelica [10], one of the leading tools in the field. However the underlying characteristics of Modelica have remained since the beginning and are still available in the outset versions of the language, precluding the related tools from matching the sizing task. Indeed since both language and computational process are simulation task oriented,

(1) only square systems of differential and algebraic equations are solved;

(2) inequalities are not taken into account during the solving process.

Item (1) implies that the parameters you need to set in order to run a calculus are precisely the design parameters you want to compute.

Item (2) implies that the solutions which are computed can be false solutions regarding the numerous technical and customer requirements which are laid down in terms of inequalities.

2.3.2. Optimizers. Amongst the commercial Analysis and Decision tools, Excel is certainly one of the most widespread. This success can be explained by the large distribution of the Office Suite in companies. In particular, numerous design engineers resort to Excel as a convenient desk calculator that offers great functionalities to set out their work results. Regarding the engineering problems the basic solving capabilities of the calculus sheets (formula evaluation and iterations for the case of circular references) are inadequate; therefore Excel provides a solver as a complementary macro.

The solver of Microsoft Excel uses 2 optimization codes:

(i) a mixed linear programming code designed and commercialized by Frontline Systems, Inc.;

(ii) a nonlinear optimization code GRG2 ("Generalized Reduced Gradient") designed by Leon Lasdon, Texas University Austin and AllanWarren, Cleveland University.

GRG2 is the code that best matches engineering problems. Unfortunately it shares the drawbacks of all iterative convergent methods: A good starting point has to be chosen in the search space and if the method converges, a local optimum is obtained. As explained above, the context of solving in a sizing task goes in the other direction: the structure of the search space is unknown and we search for globally optimal solutions.

This kind of algorithm is available in the Matlab Optimization toolbox and is the most commonly used by engineers too. The case of stochastic algorithms like Genetic Algorithms available in Matlab too will be discussed later in the fourth part of this paper.

Apart from mathematical programming methods, namely, linear programming or gradient based nonlinear optimization, other types of numerical methods can be investigated to solve engineering sizing problems. We will focus hereafter on the use of constraint programming methods and more specifically of interval constraint programming. Constraint programming is a solving paradigm which combines an original solving approach with a problem description model called constraint satisfaction problem (CSP).

3. Constraint Solving Approach

Unlike iterative methods which start from an initial point and then build series of points converging towards a solution, the solving approach is based on the idea of exploring the whole search space. This approach has been initially developed for solving combinatorial problems [11]. Since these problems have a finite search space, it is theoretically possible to generate (i.e., enumerate) each potential solution and test whether or not it is a solution or not. This general "Generate and Test" method naturally leads to a first solving algorithm scheme which is in fact an exploration algorithm. Unfortunately this algorithm is computationally intractable whenever the size of the finite search space is huge. For this reason, the constraint programming community has developed numerous sophisticated methods for reducing the search space in order to avoid costly space explorations. The more advanced solving methods combine exploration methods and reduction methods. Developing a reduction method requires defining and maintaining the variation domain of all problem variables. Domains are reduced by various consistency methods which eliminate sets of values from domain variables which are inconsistent with the problem description [12]. Reasoning on variation domains in order to approximate the set of solutions characterizes the constraint solving approach.

3.1. Constraint Satisfaction Problem (CSP).

The problem description model is called CSP in constraint programming. A CSP [13] is usually defined by a triplet <X, D, C> where X is a set of variable, D is a set of domains, and C is a set of constraints and

(i) $\forall x_i \in X$, D_i is the domain of possible values of x_i;

(ii) $\forall c_i \in C$, c_i is a constraint expressed as a relation between $\{x_j\} \subseteq X$.

A relation should be any kind of mathematical linear or nonlinear equations, inequalities, logical formulas, and so forth.

As an example, let us study the following short CSP:

$$\begin{aligned} X &= \{x1, x2\} \\ D &= \{\{0,1\},\{0,1\}\} \\ C &= \{x1 \neq x2, x1 < x2\} \end{aligned} \tag{1}$$

It is convenient to use the additional notations $D = \{D_{x1}, D_{x2}\}$; hence,

(i) $D_{x1} = \{0, 1\}$;

(ii) $D_{x2} = \{0, 1\}$.

A solution of a CSP is an assignment of all variables such that all constraints are satisfied. For instance, the unique solution of the above CSP is

$$x1 = 0;$$
$$x2 = 1.$$

This solution can be interpreted as an elimination of values from domain variables:

(i) Initially $D_{x1} = \{0, 1\}$, $D_{x2} = \{0, 1\}$.

(ii) Finally $D_{x1} = \{0\}$, $D_{x2} = \{1\}$.

This global domain reduction proceeds from the application of CSP constraints which implement a local way to reduce the values of the variables. Following the current CSP example,

(i) from $x1<x2$, we eliminate 1 from D_{x1} since there is no value of D_{x2} consistent with and then D_{x1} becomes $\{0\}$;

(ii) from $x1\neq x2$, we obviously eliminate 1 from D_{x2} and then D_{x2} becomes $\{1\}$.

After the reduction process, we have to enumerate the reduced search space. In our example, it is easy to check that the unique potential solution <x1, x2> = <0, 1> is a solution.

At this stage, it is worth noticing that a CSP is fundamentally a logical model expressed as a set of logical properties. In particular in a CSP model,

(i) the order of constraints does not matter;

(ii) constraints are noncausal properties;

(iii) solving a CSP offers some logical guarantees such as proving there is no solution, finding all solutions or finding globally optimal solutions.

On the one hand, constraint programming provides a natural way of writing problem descriptions without requiring any kind of rewriting.

On the other hand, the overall impact of constraint solving naturally suits the sizing task, i.e., finding all solutions in the design space and globally optimizing the design.

However, we have to keep in mind that this global nature has a negative effect on the computational complexity of calculus methods. In particular, constraint solving methods are much more time and memory consuming than local iterative convergent methods. Therefore, the applicability of such methods on real engineering problems remains to prove since most papers focus on short scale problem solving [14].

Finally, it is important to point up that the constraint programming framework can be adapted to various calculus domains like finite, Boolean, rational, or real domains.

3.2. Interval Constraint Programming. When real variables are considered, the domain is called interval constraint programming. The global approach remains the same as in constraint programming but domains are replaced by intervals and constraints and therefore domain reductions are developed from methods based on Interval Analysis [15] while search space methods are based on subboxes or subpavers exploration. The resulting interval solvers aim at finding all solutions of sets of nonlinear equations and inequalities or at finding a global optimum. This is called a Numerical Constraint Satisfaction Problem (NCSP).

Various tools are completely dedicated to NCSP solving like CLP($\mathfrak{R}$), UniCalc, RealPaver [16], Numerica [17], or Constraint Explorer [6]. Others include numerical features in more general CSP frameworks like Eclipse [18] or ILOG Solver C++ library [19].

We have decided to choose Constraint Explorer for our case study. The main reason for this choice is that unlike most of the other available tools which are general numerical solvers, Constraint Explorer has been specified with the aim of offering a design tool. This has led to software development presenting some characteristics which are relevant for our modelling and optimizing purposes and will be laid down hereafter.

4. Constraint Explorer

4.1. Overview. Constraint Explorer [6] is a software tool which is the major result of the CO2 project, a research project granted by the French Ministry of Research in 2002. It has been specified for modelling, sizing, and decision making in design.

Constraint Explorer addresses design problems that can be expressed by a set of relations on integer or real variables. These relations are either equations or inequalities; they can be linear or nonlinear. Expert rules supplement the language in order to dynamically store in the model additional relations which are expressed in the right-hand side of the rules and activated according to the logical state of left-hand side conditions.

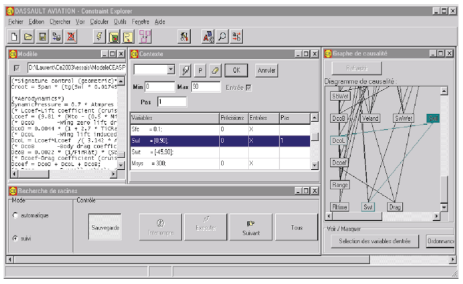

Figure 1: Constraint Explorer Front view.

Constraint Explorer supplies numerical algorithms based on solving methods supplied by the numerical constraint programming community.

Most of the functionalities are available through a graphical user interface (cf. Figure 1).

4.2. Modelling with Constraint Explorer. As explained above, the language for modelling a CSP in constraint programming is the <V, D, C> model. The first experiments with various engineering problems have pointed out that this model was too poor for efficiently describing industrial design problems. If we make a kind of comparison between constraint programming and classical programming, we could say that <V, D, C> is closer from a low-level machine language than from a high-level end-user language. The Constraint Explorer language has extended this model in order to increase the power of expression and then to make the design problem solving easier. The most important features are

(i) the introduction of constants as primitive elements of the model like variables or constraints;

(ii) many types of variables depending on the nature of their variation domain: real, integer, enumerated integer, and enumerated real;

(iii) declaration of functions, e.g., f(_x,_y) ≔ (_x+_y)/2;

(iv) declaration of aliases, e.g., $\#m_{ab} \coloneqq f(a, b)$;

(v) declaration of matrix and matrix calculus.

All these features help to describe mathematical formulas intensively used in engineering design problems.

For instance, the following expression

$$res = \left(PKe \times \begin{bmatrix} \sigma^2 \\ \sigma^1 \\ \sigma^0 \end{bmatrix} \right) \times \begin{bmatrix} (\ln(\mathrm{Re}) + \mathrm{Re})^2 \\ (\ln(\mathrm{Re}) + \mathrm{Re})^1 \\ (\ln(\mathrm{Re}) + \mathrm{Re})^0 \end{bmatrix}^T \quad (2)$$

is naturally expressed in the Constraint Explorer modelling language:

```
f(_X)=ln(_X)+_X
#d = f(Re);

PKe:mat[3,3] = [[−3.05762874E − 04,...]

M:mat[3,1] = [[s^2],[s],[1]] ;
N:mat[3,1] = [[#d^2],[#d],[1]] ;
res:mat [3,3] ;




res=(PKe x M) x trans(N);
```

Constraints building can use all these elements either to express equations or inequalities built from sophisticated mathematical expressions or to express n-ary user constraints built from an extension table of valid n-tuples, i.e., the set of all consistent values.

Moreover, additional syntactic sugars like n-ary sums and products operators for expressions, iterators for constraints, or meta-constraints like rules complete the language and still increase its modelling capabilities.

4.3. Solving with Constraint Explorer. The solving techniques implemented in CE rely on a branch and prune algorithm. Pruning is provided by recent interval constraint consistency techniques (namely, Hull and Box consistency [20]). Branching allows you to explore the search space by bisecting the variable domains.

We can roughly distinguish three steps in the branch and prune algorithm:

(1) Application of contractors (or reduction operators) that prune the domain variables over a single constraint.

(2) Propagation of the domain reduction from constraint to constraint until a fixed point on intervals is reached.

(3) Application of a bisection algorithm to exhaustively explore the design space.

Let us describe the branch and prune algorithm on a system with a nonlinear constraint:

$$\begin{aligned} x + y &= 50 \\ x * y &= 400 \end{aligned} \tag{3}$$

Starting from $x, y \in [0, 100]$, we reduce from the first equation x and y. Indeed since $x + y = 50$ and $x > 0$ and $y > 0$ then necessarily $x, y \in [0, 50]$. Then from the second equation, we can infer that all values of x or y smaller than 8 are to be discarded ($x \in 400 /[0, 50]$). Again in the first equation, we derive the newly reduced domain for x and y: $[8 \ldots 42]$ and so on and so forth. When no more variations are possible (or they become too small to be considered) we have reached a fixed point (here $x, y \in [10, 40]$). If the domain length is greater than the required precision, we bisect the domain and start a similar domain reduction on the left and right subproblems. The algorithm ends when no more bisection is possible. In our example, the left subproblem ($x \in [10, 25]$) leads to the first solution $x = 10$, $y = 40$ and the right subproblem ($x \in [25, 40]$) leads to the second solution $x = 40$, $y = 10$.

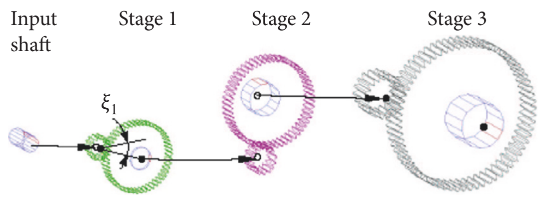

Figure 2: A three-stage transmission mechanism.

This example illustrates the essential property of this algorithm which is called completeness: no solution of the problem is lost.

5. Application to a Power Transmission System

5.1. System Description. Let us apply those concepts, methods, and tools to our test case: the sizing of a three-stage power transmission mechanism. The goal of such a mechanism is to transmit a power and modify the ratio between the input speed and the output one (this is called the reduction ratio). This object should be modelled as a system. It can be decomposed into an input shaft and three interrelated subsystems called stages. Each of those subsystems is an aggregation of three interrelated parts (or components) (cf. Figure 2).

5.1.1. Overall Description. As we said previously, a multistage transmission mechanism is a complex system. It should be split into an input shaft and several stages. Each stage should be first modelled as a component and interactions between components have to be expressed. Finally, global relations and geometrical (closure conditions) as much as technological ones (required reduction ratio) have to be modelled.

5.1.2. Design Variables. Each stage of the transmission system is defined by seven design parameters: an angle, a module, the tooth number of the pinion, the tooth number of the wheel, a shaft radius, and a shaft length (Table 2). As soon as the 3 x 7=21 Design Variables are valued under constraints, the system is fully defined. We should add two other Design Variables, i.e., the radius and the length of the input shaft of the mechanism.

5.1.3. One-Stage Description. As we mentioned above, each stage has three elements: a pinion, a wheel, and an output shaft. This model includes local level constraints such as the equations describing each stage or each gear. We should find in annex A the nomenclature of the product.

Table 2: The design variables.

Input Shaft	Angle	Module	Tooth number		Face width	Shaft radius $r_{a,0}$	Shaft length $l_{a,0}$
			Pinion	Wheel			
Stage 1	ξ_1	m_1	$Z_{1,1}$	$Z_{2,1}$	b_1	$r_{a,1}$	$l_{a,1}$
Stage 2	ξ_2	m_2	$Z_{1,2}$	$Z_{2,2}$	b_2	$r_{a,2}$	$l_{a,2}$
Stage 3	ξ_3	m_3	$Z_{1,3}$	$Z_{2,3}$	b_3	$r_{a,3}$	$l_{a,3}$

Let us recall the constraints of one stage [21]:

(i) Acceptable maximum transmission powers with the surface pressure and the rupture of the teeth:

$$P_{\text{mini}} \leq \left(\frac{C_1 \cdot C_2 \cdot C_3 \cdot C_4 \cdot C_5 \cdot C_6}{K_{B_p}}\right) \tag{4}$$

$$P_{\text{mini}} \leq \left(\frac{C_{B1} \cdot C_{B2} \cdot C_{B3} \cdot C_{B4} \cdot C_{B5} \cdot C_{B6}}{K_{B_R}}\right) \tag{5}$$

We should find in Appendix B the expression of $C_1, \ldots, C_6$ and $C_{B1}, \ldots, C_{B6}$.

(ii) Condition on the transverse contact ratio:

$$1.3 \leq \varepsilon_{\alpha,s} \tag{6}$$

(iii) Condition on the linear velocity of teeth:

$$\frac{Z_{1,s}.V}{100}\sqrt{\frac{Z_{2,s}^2}{Z_{2,s}^2 + Z_{1,s}^2}} \leq 10 \tag{7}$$

(iv) Conditions on meshing interference:

$$\pi.Y_{1,s}.U_{1,s} \leq \frac{Z_{2,s}}{2}.\tan\alpha_t' \tag{8}$$

$$\pi.Y_{2,s}.U_{2,s} \leq \frac{Z_{1,s}}{2}.\tan\alpha_t' \tag{9}$$

(v) Limit on the value of the face width compared to the diameter of pinion and wheel:

$$b_s \leq d_{1,s} = m_s Z_{1,s} \tag{10}$$

$$b_s \geq 0.1 d_{2,s} = 0.1 m_s Z_{2,s} \tag{11}$$

$$\max\left\{d_{1\text{Mini},s}, \frac{d_{2\text{Mini},s}}{\overline{u}_s}\right\} \leq d_{1,s} \tag{12}$$

$$d_{1,s} \leq \min\left\{d_{1\text{Maxi},s}, \frac{d_{2\text{Maxi},s}}{\overline{u}_s}, \frac{60V_{\text{Max}}}{\pi.N_s}\right\} \tag{13}$$

$$b_s \leq \min\left\{d_{1,s}, b_{\text{Maxi},s}\right\} \leq 0 \tag{14}$$

$$\max\left\{0, 1.d_{2,s}, b_{\text{Mini},s}\right\} \leq b_s \tag{15}$$

(vi) Noninterference between shaft, pinion, and wheel (cf. Figure 3):

$$\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2 > \left(r_i + r_j\right)^2$$
$$\left|\left(z_i + \frac{h_i}{2}\right) - \left(z_j + \frac{h_j}{2}\right)\right| > \frac{h_i + h_j}{2} \tag{16}$$

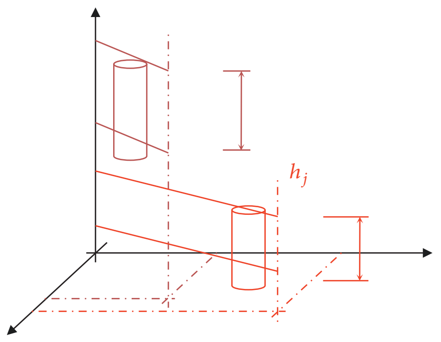

Figure 3: Noninterferences between shaft and gears.

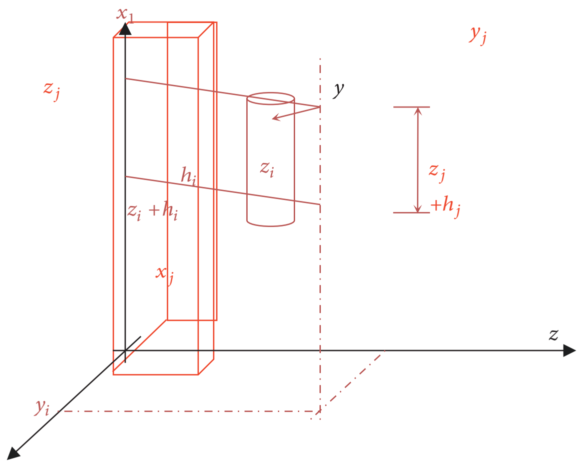

Figure 4: Noninterferences with the casing box.

(vii) Noninterference with the casing box (cf. Figure 4):

$$\begin{gathered}\left(x_i - r_i > x_{min}\right), \\ \left(x_i + r_i < x_{max}\right) \\ \left(y_i - r_i > y_{min}\right), \\ \left(y_i + r_i < y_{max}\right) \\ \left(z_i > z_{min}\right), \\ \left(z_i + h_i < z_{max}\right)\end{gathered} \tag{17}$$

5.1.4. Connexions between Stages. Let us now recall the constraints related to the interactions between stages [18]:

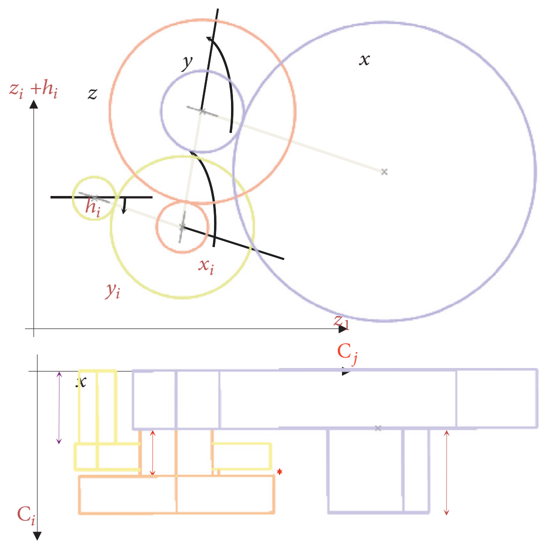

Figure 5: Closure conditions.

(i) Torsion shaft resistance between stages:

$$\frac{2.10^3 . C_{e,s} . (Z_{1,s}/Z_{2,s})}{G . \theta_{max} . \pi . r_{a,s}{}^4} \leq 1 \tag{18}$$

(ii) Consistency between velocities:

$$N_{2,s} = N_{1,s-1} \tag{19}$$

Geometrical Consistency and Global Relations

(i) Speed ratio:

$$\left| u_r - \prod_{s=1}^{s=3} \frac{Z_{2,s}}{Z_{1,s}} \right| \leq \Delta u_r \tag{20}$$

(ii) Closure conditions (cf. Figure 5):

$$Z_I + (r_{1,1} + r_{2,1}) . \sin(\xi_1) + (r_{1,2} + r_{2,2}) . \sin(\xi_1 + \xi_2) + (r_{1,3} + r_{2,3}) . \sin(\xi_1 + \xi_2 - \xi_3) = Z_O \tag{21}$$

$$Y_I + (r_{1,1} + r_{2,1}) . \cos(\xi_1) + (r_{1,2} + r_{2,2}) . \cos(\xi_1 + \xi_2) + (r_{1,3} + r_{2,3}) . \cos(\xi_1 + \xi_2 - \xi_3) = Y_O \tag{22}$$

$$X_I + l_{a,0} + b_1 + l_{a,1} - l_{a,2} + l_{a,3} = X_O \tag{23}$$

5.1.5. Design Requirements. The main requirements are to size the components and to set them together in order to transmit a given power in a given reduction ratio u_r with a tolerance Δu_r and minimizing the space required for the system in overall dimensions. Obviously we need to satisfy the power transmission P_{min}, the reduction ratio u_r, and the tolerance Δu_r requirements.

5.1.6. Optimizing the Volume of the System. The goal is to minimize the overall volume of the mechanism defined as follows:

$$f_{Obj}(\mathbf{x}) = \pi . \sum_{s=0}^{s=3} \left(r_{a,s}{}^2 . l_{a,s} \right) + \pi . \sum_{s=1}^{s=3} \left(b_s . m_s{}^2 . \left(Z_{1,s}{}^2 + Z_{2,s}{}^2 \right) \right) \tag{24}$$

5.2. Problem Modelling with Constraint Explorer. All the constraints have been directly expressed in the Constraint Explorer language and the resulting model did not require any kind of formal rewriting before being compiled and solved. Once compiled with Constraint Explorer, the various primitive objects of the computational model can be numbered. The results are summarized hereafter:

(i) 57 constants

(ii) 112 variables and amongst them

 (a) 23 Design Variables

 (b) 89 Behaviour Variables

(iii) 1 performance indicator

As explained in the previous Constraint Explorer description paragraph, various types of domain variables are used in the current model: 6 integer variables, 3 enumerated real variables, and 103 pure real variables.

All these variables are connected through the set of equations and inequalities of the model. The model is composed of

(i) 89 equations

(ii) 48 inequalities

(iii) 1 optimization criterion

The following figure details a part of the Constraint Explorer Three-Stage Design model. It concerns the conditions on the linear velocity of teeth and on meshing interference. In this model, "_e" is an index iteratively representing the three different stages of the transmission system; therefore there are nine inequalities.

```
for(_e,1,3) {
(∗ Condition on the linear velocity of teeth ∗)
Z[1,_e]∗V(Z[1,_e], N1[1,_e], mn0[1,_e])∗
sqrt(Z[2,_e]^2/(Z[2,_e]^2 + Z[1,_e]^2)) <= 1000;
(∗ Conditions on meshing interference ∗)
pi∗Y1(mn0[1,_e],mt0[1,_e])∗U1(Z[1,_e],
alphaprimt[1,_e],Y1(mn0[1,_e],mt0[1,_e]))<=Z[2,_e]
∗tan(alphaprimt[1,_e])/
pi∗Y1(mn0[1,_e],mt0[1,_e])∗U2(Z[2,_e],
alphaprimt[1,_e],Y1(mn0[1,_e],mt0[1,_e]))<=Z[1,_e]
∗tan(alphaprimt[1,_e])/
}
```

The comparison of this model with the related mathematical inequalities described beforehand illustrates the fact that the translation effort is minimized.

5.3. Problem Solving with Constraint Explorer. Once modelled, the problem remains to be solved. As explained above, the solving method of Constraint Explorer is based on a generic branch and prune algorithm that combine space search reduction methods (pruning) with space search exploration methods (branching). The space search exploration consists in bisecting variables whose domain is not reduced to a value. It is well established that the choice of the variables to bisect may have a major impact on the performance of the branch and prune algorithm [22]. In order to improve its performance, Constraint Explorer may propose various heuristics for ordering variables which do not fall into the scope of the present paper and therefore will not be described here. However concerning the current power transmission problem, one must underline that all results and performances which are presented hereafter are issued from a static variables ordering that makes use

(1) of the system architecture: in particular it is stage composition;

(2) of the type of the variables: integer variables are bisected before enumerated variables and enumerated variables are bisected prior to real variables.

Therefore in a given stage, the pinion tooth number, the wheel tooth number, and the module value are bisected in this order.

5.4. Results and Validation. We would now like to illustrate our approach with a set of numerical results. The initial domains given by the expert for the constraint variables are presented in Table 3 column 1.

After adding the following requirements,

$$P_{min} = 8.8 \text{ kW}$$
$$N_{Input} = 1500 \text{ rpm}$$
$$u_r = 44$$
$$\Delta u_r = 0.01,$$

and after the first propagation, the intervals are reduced as in Table 3 column 2.

The solving process (i.e., a branch and prune mechanism) ends on a first admissible solution presented in Table 3 column 3. The last step gives an optimal solution as shown in Table 3 column 4.

As a result, we found an optimal solution with $f_{obj} = 8.1 10^6$ mm^3. Calculations took a computing time of two-and-half-minute computing time on a Intel(R) Pentium(R) M 1.6GHz 512Mo RAM and a process time of 50 seconds on a Dell Optiplex 740 AMD Athlon Dual Core Processor 5400B 2.81 Ghz, 1.93 Go de RAM.

We checked the validity of those results in two ways:

(i) We implemented a dedicated C code to measure the differences between the values of the design and performance variables given by CE and those processed in the C program by the instantiated formulas. The maximum difference is approximately 1e-12, which means that CE seems to be quite a robust solver.

(ii) We built a working drawing to check the closure and noninterference conditions of the instantiated mechanism.

5.5. Comparing with Other Approaches. Usually, a constrained optimization problem on mix variables as the power transmission one is modelled as follows:

$X = (x_1, x_2, \ldots, x_n)$ a set of n variables. The value of each x_i should be in $\mathbb{R}$ or in $\mathbb{Z}$.

$$f : \mathbb{R}^r \times \mathbb{Z}^{n-r} \longrightarrow \mathbb{R} \qquad \text{a scalar function to minimize} \tag{25}$$

$$\begin{gathered}\text{such that}\\ \forall i \in \{1, \ldots, p\},\\ \mathcal{X} \subseteq X,\\ g_i(\mathcal{X}) = 0 \qquad \text{a set of p equations to satisfy}\\ \forall j \in \{1, \ldots, q\},\\ \mathcal{X}' \subseteq X,\\ h_j(\mathcal{X}') \le 0 \qquad \text{a set of q inequalities to satisfy}\end{gathered} \tag{26}$$

Two main categories of techniques for finding a global minimum to f are described in technical literature:

(i) The deterministic methods: as soon as f, g_i, and h_j are linear functions, Mix Integer Linear Programming (MILP) algorithms [23, 24] can be used to find the global optimum. Moreover, when f is a quadratic function, the right way is to use Mix Sequential Quadratic Programming (SQP) algorithms [25]. Unfortunately a lot of functions are nonlinear in the engineering field. MLP and SQP should take account of such functions by increasing the number of variables in the problem. On the one hand, however, this increases the complexity while users, on the other hand, have to make specific transformations to set a nonlinear problem into a linear or a quadratic one if possible. Thus, particularly in the case of design problems, the structure of the initial model (i.e., the nonlinear equations and inequalities) is lost.

(ii) The stochastic methods: they draw upon nature and are called Genetic Algorithms (GA) [26–28], Simulated Annealing (SA) [29, 30], or Particle Swarm Optimization (PSO) [31, 32]. All of them give an approximation of the global optimum. Optimization problems in design have been mainly solved by GA algorithms [27, 28].

In technical literature, the main approach to design problem solving is the evolutionary (genetic) algorithm method.

Table 3: Results.

Variables	Initial Domains	First Propagation	Admissible solution	Optimal solution
		Design Variables (DVs)		
Input Shaft				
$r_{a,0}$ (mm)	$[10.0, 507.073]$	$[12.937, 503.942]$	12.937	12.937
$l_{a,0}$ (mm)	$[10.0, 1014.15]$	$[10.0, 1014.15]$	10.155	10.0
Stage1				
m_1	$[0.5, 50.0]$	$[0.5, 22.0]$	1.5	1.5
$Z_{1,1}$	$[11, 45]$	$[11, 45]$	14	14
$Z_{2,1}$	$[20, 150]$	$[20, 150]$	20	20
b_1 (mm)	$[10, 500]$	$[10, 330]$	10.0	10.0
$r_{a,1}$ (mm)	$[10.0, 507.073]$	$[10.0, 503.875]$	10.0	10.0
$l_{a,1}$ (mm)	$[10.0, 1014.15]$	$[10.0, 1014.15]$	19.1168	84.9537729
ξ_1 (rad)	$[-\pi, \pi]$	$[-\pi, \pi]$	-0.1533	-0.058
Stage 2				
m_2	$[0.5, 50.0]$	$[0.5, 22.0]$	5.5	6.0
$Z_{1,2}$	$[11, 45]$	$[11, 45]$	16	17
$Z_{2,2}$	$[20, 150]$	$[20, 150]$	56	60
b_2 (mm)	$[10, 500]$	$[10, 330]$	14.0	10.0
$r_{a,2}$ (mm)	$[10.0, 507.073]$	$[10.0, 503.875]$	10.0	10.0
$l_{a,2}$ (mm)	$[10.0, 1014.15]$	$[10.0, 1014.15]$	10.0	10.0
ξ_2 (rad)	$[-\pi, \pi]$	$[-\pi, \pi]$	-0.0169	
Stage 3				
m_3	$[0.5, 50.0]$	$[0.5, 22.0]$	2.75	2.25
$Z_{1,3}$	$[11, 45]$	$[11, 45]$	17	17
$Z_{2,3}$	$[20, 150]$	$[20, 150]$	149	149
b_3 (mm)	$[10, 500]$	$[10, 330]$	16.0	10.0
$r_{a,3}$ (mm)	$[10.0, 507.073]$	$[10.0, 503.875]$	10.0	10.0
$l_{a,3}$ (mm)	$[10.0, 1014.15]$	$[10.0, 1014.15]$	189.22	123.5182346
ξ_3 (rad)	$[-\pi, \pi]$	$[-\pi, \pi]$	-0.4	-0.09
		Performance Indicator		
f_{obj} (mm^3)	$[0, +\infty]$	$[24842.14, 20821e7]$	13194785.4	**8.1 e6**

Fauroux and Lafon [18] implemented and applied a Genetic Algorithm to solve the problem addressed in this paper.

Although Genetic Algorithms are actually most useful we would like to focus on two key points: On the one hand, the capability to find the global optimum depends on the crossing operators used, the frequency of mutation tuning, and the simulation duration. On the other hand, the g_i and h_j functions are mainly integrated as penalty functions (i.e., within the objective function) and in this case again, the structure of the design problem is lost.

Moreover, in such an approach, it is necessary to study the set of equations and inequalities of the problem to minimize the number of variables by doing substitutions of variables and rewriting of expressions. This is not the case with the CSP approach because the lack of a declarative and modular modelling language negatively impacts software maintenance and upgrading. As a consequence, the capability of models capitalization is very weak.

Although stochastic methods seem to be easy to program, in counterpart the computational performance is poor and the related optimization process is nonglobal with a reasonable finite time.

By way of example, [18] implements a final objective function as follows:

$$f_{Evaluation} = f_{obj} + r \times (Sum\ of > 0\ violated\ relations) \tag{27}$$

where r is a penalty factor, progressively increased during the resolution.

Moreover, the system of equations and inequalities is simplified manually. The aim is to remove all the Behavioural Variables from the problem. As a result, the authors found an optimal solution with $f_{obj} = 3.74\ 10^7$ mm^3. It took 45 min computer time to work out the calculations on a HP PA-RISC 8600 750 MHz.

6. Conclusion

We have shown that specifying a design sizing problem is possible with a model-based approach using constraint solving technology.

Our proposal consists in two main points:

A specification of system sizing problem based on a dedicated terminology: Design Variables, Behaviour Variables, and Performance Indicators.

An efficient modelling and solving process combining model-based approach and constraint solving paradigm.

The application of our model-based approach to a power transmission system has shown that it is possible to successfully apply this approach to a scalable problem and with good results. More precisely, they are better than those obtained with a stochastic one (i.e., with Genetic Algorithm) in terms of both performance and quality. Moreover the problem model does not require any kind of tedious rewriting. Requirements are naturally expressed and added to the model as inequalities.

Constraint modelling gives us a chance to start defining a language accounting for both system description and elicitation of needs. Nevertheless, there remains a gap to address engineering needs. Even if equations and inequalities are directly translated, the structure of the system (i.e., components structure and relations between components and subsystems) is lost. A work is in progress to define a new higher level structured modelling language for design problem specifications.

Appendix

A. Nomenclature of Parameters and Variables

Requirements (3)

N_{Input}: Input rotational speed

P_{Min}: Minimal required input power

u_r: Global required reduction ratio of the speed reducer

Δu_r: Tolerance on required global ratio.

Design Variables (DVs) (23)

m_s: Tooth real module of stage s (1,2,3)

$Z_{i,s}$: Number of teeth for wheel i of stage s

$l_{a,s}$: Length of input shaft (0) or output shaft of stage s (1,2,3)

$r_{a,s}$: Radius of input shaft (0) or output shaft of stage s (1,2,3)

b_s: Gear face width of stage s (1,2,3)

ξ_s: Angular self rotation of stage s (1,2,3) along its input shaft.

Behavioural Variables (BVs)

s: Index stage number

C_i: Contact stress factors

C_1: Speed and ratio coefficient, nonlinear function of m_s, $Z_{1,s}$, and $Z_{2,s}$

C_2: Tooth shape coefficient depends on α_n and β

C_3: Speed coefficient, nonlinear function of m_s, $Z_{1,s}$, $Z_{2,s}$

C_4: Load distribution coefficient, nonlinear function of m_s, $Z_{1,s}$, $Z_{2,s}$, b_s

C_5: Material coefficient considered as a constant value

C_6: Tooth contact coefficient considered as a constant value

C_{Bi}: Bending stress factors

C_{B1}: Speed and ratio coefficient, nonlinear function of m_s, $Z_{1,s}$, and $Z_{2,s}$

C_{B2}: Contact and overlap ratio coefficient considered as a constant value

C_{B3}: Dynamic behaviour coefficient, nonlinear function of m_s, $Z_{1,s}$, $Z_{2,s}$

C_{B4}: Shape and constraint coefficient, nonlinear function on $Z_{1,s}$, $Z_{2,s}$

C_{B5}: Load distribution coefficient, depending on ms, $Z1,s$, $Z2,s$, bs

C_{B6}: Fatigue stress coefficient, depending on material and considered as a constant value

C_{B7}: Stress concentration coefficient, depending mainly on m_s

$d_{i,s}$: Diameter for pinion (i=1) or wheel (i=2) of stage s

K_{BP}, K_{BR}: Service factor for contact / bending stress, supposed as a constant value

$U_{i,s}$: Geometry coefficient for wheel i, nonlinear function of m_s, $Z_{i,s}$

V: Linear velocity on teeth

V_{max}: Maximal allowed linear velocity (typically 20m/s)

$Y_{i,s}$: Geometry coefficient for wheel i, nonlinear function of m_s, $Z_{i,s}$

α_n, α_t': Normal and working transverse pressure angle

β: Reference helix angle

$\varepsilon_{\alpha,s}$: Transverse contact ratio, nonlinear function of m_s, $Z_{1,s}$, $Z_{2,s}$

$C_{e,s}$: Input torque at stage s

G: Shaft shearing modulus

O_i: Input shaft location on the housing

O_o: Output shaft location on the housing

ri, s: Radius of wheel i of stage s

θ_{Max}: Maximum torsion angle for shafts (typically 0.1°/m).

Performance Indicators (BIs) (1)

f_{obj}: Overall volume of the mechanism.

B. Equations and Inequalities

(i) C_1 factor:

$$C_1 = \frac{\pi}{60} N_1 \left(\frac{Z_1}{Z_1 + Z_2} \right) \quad \text{(B.1)}$$

(ii) C_2 factor:

$$C_2 = \frac{1}{Z_H^2 Z_\varepsilon^2 Z_\beta^2} \quad \text{(B.2)}$$

$$Z_\beta^2 = \cos\beta_0 \quad \text{(B.3)}$$

$$Z_H^2 = \frac{2\cos\beta_0 \cos\alpha_{n0} \cos\alpha_t'}{\cos\alpha_{t0}{}^3 \sin\alpha_t'} \quad \text{(B.4)}$$

$$Z_\varepsilon^2 = \begin{cases} \dfrac{1}{\varepsilon_\alpha} & si\ \varepsilon_\beta > 1 \\ \dfrac{4-\varepsilon_\alpha}{3}\left(1-\varepsilon_\beta\right) + \dfrac{\varepsilon_\beta}{\varepsilon_\alpha} & si\ \varepsilon_\beta < 1 \end{cases} \quad \text{(B.5)}$$

$$\varepsilon_\beta = \frac{b.\sin\beta_0}{\mu.m_{n0}} \quad \text{(B.6)}$$

(iii) C_3 factor:

$$C_3 = \frac{Z_v^2}{K_v} \quad \text{(B.7)}$$

$$Z_v^2 = C_{zv} + \frac{2\left(1-C_{zv}\right)}{\sqrt{0.8 + 32/V}} \quad \text{(B.8)}$$

$$C_{zv} = 0.85 - \frac{0.85\left(\sigma_{Hlim} - 850\right)}{350} \quad \text{(B.9)}$$

$$\text{Si } \sigma_{Hlim} < 850MPa \text{ alors } \sigma_{Hlim} = 850\ MPa \quad \text{(B.10)}$$

$$\text{Si } \sigma_{Hlim} > 850MPa \text{ alors } \sigma_{Hlim} = 1200\ MPa \quad \text{(B.11)}$$

$$V = \frac{\pi.m_{n0} Z_1 N_1}{60.10^3 \cos\beta_0} \quad \text{(B.12)}$$

$$K_v = 1 + \left[\frac{K_1}{K_A\left(F_t/b\right)} + K_2 \right] . \frac{Z_1.V}{100} . \sqrt{\frac{u^2}{u^2+1}} \quad \text{(B.13)}$$

$$u = \frac{Z_2}{Z_1} \quad \text{(B.14)}$$

$$F_t = \frac{60.10^6 P_{mini} \cos\beta_0}{\pi.m_{n0} Z_1 N_1} \quad \text{(B.15)}$$

(iv) C_4 factor:

$$C_4 = \frac{10^{-6} b \left(m_{t0} Z_1\right)^2}{K_{H\beta} K_{H\alpha}} \quad \text{(B.16)}$$

$$K_{H\beta} = A_1 + B_1 \left(\frac{b}{m_{t0} Z_1} \right)^2 + C_1.b \quad \text{(B.17)}$$

$$K_{H\alpha} = D_1 \quad \text{(B.18)}$$

(v) C_5 factor:

$$C_5 = \frac{\sigma_{Hlim}{}^2}{Z_E{}^2} \quad \text{(B.19)}$$

$$Z_E{}^2 = \sqrt{\frac{1}{\pi\left(\left(1-\nu_1{}^2\right)/E_1 + \left(1-\nu_2{}^2\right)/E_2\right)}} \quad \text{(B.20)}$$

$$\sigma_{Hlim} = \min\left(\sigma_{Hlim1}, \sigma_{Hlim2}\right) \quad \text{(B.21)}$$

(vi) C_6 factor:

$$C_6 = Z_L^2.Z_R^2.Z_W^2 \quad \text{(B.22)}$$

$$Z_R = \left(\frac{3}{R_m} \right)^{0.12} \quad \text{(B.23)}$$

$$\text{If } Z_W \le 400\ HB \quad \text{then } Z_W = \left(1.2 - \frac{H_{Broue} - 130}{1700}\right) \quad \text{(B.24)}$$

$$\text{If } H_{broue} > 400\ HB \text{ then } Z_W = 1 \quad \text{(B.25)}$$

$$Z_L = 1 \quad \text{(B.26)}$$

$$K_{Bp} = \frac{K_A.K_R}{Z_N} \quad \text{(B.27)}$$

(vii) C_{B1} factor:

$$C_{B1} = \frac{\pi}{60} 10^6 N_1 m_{n0}{}^2 Z_1 \quad \text{(B.28)}$$

(viii) C_{B2} factor:

$$C_{B2} = \frac{1}{Y_\varepsilon Y_\beta \cos\beta_0} \quad \text{(B.29)}$$

$$\text{If } \beta_0 \le 30^\circ \text{ then } Y_\beta = 1 - \min\left(1, \varepsilon_\beta\right) \frac{\beta_0}{120} \quad \text{(B.30)}$$

$$\text{If } \beta_0 > 30^\circ \text{ then } Y_\beta = 1 - 0.25\varepsilon_\beta \quad \text{(B.31)}$$

$$Y_\varepsilon = 0.25 + \frac{0.75}{\varepsilon_\alpha} \quad \text{(B.32)}$$

(ix) C_{B3} factor:

$$C_{B3} = \frac{1}{K_V} \quad \text{(B.33)}$$

(x) C_{B4} factor:

$$C_{B4} = \frac{1}{Y_{FA}Y_{SA}} \tag{B.34}$$

$$Y_{FA} = \frac{6h_{FA}\cos\alpha_a}{S_{Fn}^2\cos\alpha_{n0}} \tag{B.35}$$

$$S_{Fn} = 2\left[\frac{Z_n}{2}\sin\left(\frac{\pi}{3}-\gamma\right) - \left(\frac{\nu_p}{\cos\gamma}+\varsigma_{a0}\right)\cos\left(\frac{\pi}{6}\right)\right] \tag{B.36}$$

$$h_{FA} = \frac{Z_n}{2}\left(\frac{\cos\alpha_{n0}}{\cos\alpha_a} - \cos\left(\frac{\pi}{3}-\gamma\right)\right) + \left(\frac{\nu_p}{\cos\gamma}+\varsigma_{a0}\right)\sin\left(\frac{\pi}{6}\right) \tag{B.37}$$

$$\nu_p = h_{a0} - \chi_1 - \varsigma_{a0} \tag{B.38}$$

$$Z_n = \frac{Z_1}{\cos\beta_0^{\,3}} \tag{B.39}$$

$$\cos\alpha_a = \frac{Z_1\cos\alpha_{t0}}{Z_1 + 2\cos\beta_0\left(1+\chi_1\right)} \tag{B.40}$$

$$s = \frac{\pi}{2} + 2\chi_1\tan\alpha_{n0} \tag{B.41}$$

$$\alpha_a = \tan\alpha_a - \frac{s}{Z_n} - \left(\tan\alpha_{n0} - \alpha_{n0}\right) \tag{B.42}$$

$$\gamma + \frac{2\nu_p}{Z_n}\tan\gamma = \frac{\pi}{3} - \frac{2H}{Z_n} \tag{B.43}$$

$$H = \frac{\pi}{4} + h_{a0}.\tan\alpha_{n0} + \varsigma_{a0}.\tan\left(\frac{\pi}{4} - \frac{\alpha_{n0}}{2}\right) \tag{B.44}$$

$$Y_{SA} = \left(1.2 + 0.13L_a\right)q_s^{t_s} \tag{B.45}$$

$$q_s = \frac{S_{Fn}}{2\varsigma_F} \tag{B.46}$$

$$L_a = \frac{S_{Fn}}{h_{FA}} \tag{B.47}$$

$$t_s = \frac{1}{\left(1.21 + 2.3/L_a\right)} \tag{B.48}$$

$$\varsigma_F = \varsigma_{a0} + \frac{2\nu_p^{\,2}}{\cos\gamma\left(Z_n\cos\gamma^2 + 2\nu_p\right)} \tag{B.49}$$

(xi) C_{B5} factor:

$$C_{B5} = \frac{b}{K_{F\beta}K_{F\alpha}} \tag{B.50}$$

$$K_{F\beta} = K_{H\beta} \tag{B.51}$$

$$K_{F\alpha} = K_{H\alpha} \tag{B.52}$$

(xii) C_{B6} factor:

$$C_{B6} = Y_{ST}\sigma_{Flim} \tag{B.53}$$

$$Y_{ST} = 2 \tag{B.54}$$

(xiii) C_{B7} factor:

$$C_{B7} = Y_{\delta relT}.Y_{RrelT}.Y_X \tag{B.55}$$

$$Y_{\delta relT} = A_2Y_{SA} + B_2 \tag{B.56}$$

$$Y_{RrelT} = A_3.B_3\left(R_m + 1\right)^{C_3} \tag{B.57}$$

$$Y_X = A_4 - B_4m_{n0} \tag{B.58}$$

$$K_{B_R} = \frac{K_AK_R}{Y_{NT}} \tag{B.59}$$

$$\frac{Z_1V}{100}\sqrt{\frac{Z_2^{\,2}}{Z_2^{\,2}+Z_1^{\,2}}} < 10 \tag{B.60}$$

$$m_{t0} = \frac{m_{n0}}{\cos\beta_0} \tag{B.61}$$

$$\tan\alpha_{t0} = \frac{\tan\alpha_{n0}}{\cos\beta_0} \tag{B.62}$$

$$\tan\alpha_t' - \alpha_t' = \tan\alpha_{t0} - \alpha_{t0} + 2.\tan\alpha_{n0}.\frac{\chi_1+\chi_2}{Z_1+Z_2} \tag{B.63}$$

$$m_t' = m_{t0}\frac{\cos\alpha_{t0}}{\cos\alpha_t'} \tag{B.64}$$

$$\varepsilon_\alpha = Y_1.U_1 + Y_2.U_2 \tag{B.65}$$

$$Y_{i=1,2} = \frac{m_{t0}Z_i + 2m_{n0}\left(1+\chi_i\right) - m_t'.Z_i}{2m_t'} \tag{B.66}$$

$$U_{i=1,2} = \frac{1}{\mu\cos\alpha_t'}\left[\sqrt{\left(\frac{Z_i\sin\alpha_t'}{2Y_i}\right)^2 + \frac{Z_i}{Y_i} + 1} - \frac{Z_i\sin\alpha_t'}{2Y_i}\right] \tag{B.67}$$

$$\pi Y_1U_1 < \frac{Z_2}{2}\tan\alpha_t' \tag{B.68}$$

$$\pi Y_2U_2 < \frac{Z_1}{2}\tan\alpha_t' \tag{B.69}$$

Data Availability

The problem we addressed is fully described inside this manuscript. That is to say, all variables, equations, and inequalities of the problem are available inside this document. Readers can use this data as they want.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] G. Pahl, W. Beitz, and K. Wallace, *Engineering Design - A Systematic Approach*, Springer, London, UK, 1996.

[2] D. C. Brown and B. Chandrasekaran, "Knowledge and Control for a Mechanical Design Expert System," *The Computer Journal*, vol. 19, no. 7, pp. 92–100, 1986.

[3] D. Ruland and T. Spindler, "Integration of product and design data using a metadata- and a rule-based approach," *Computer Integrated Manufacturing Systems*, vol. 8, no. 3, pp. 211–221, 1995.

[4] Z. Shi, H. Zhou, and J. Wang, "Applying case-based reasoning to engine oil design," *Artificial Intelligence in Engineering*, vol. 11, no. 2, pp. 167–172, 1997.

[5] P. A. Yvars, P. Lafon, and L. Zimmer, "Optimization of mechanical system: Contribution of constraint satisfaction method," in *CIE'39, International Conference on Computers and Industrial Engineering*, pp. 1379–1384, Troyes, France, July 2009.

[6] L. Zimmer and P. Zablit, "Global aircraft predesign based on constraint propagation and interval analysis," in *Proceedings of CEAS Conference on multidisciplinary Aircraft design and Optimisation*, Köln, Germany, 2001.

[7] D. G. Ullman, *The Mechanical Design Process*, McGraw-Hill, New York, NY, USA, 1997.

[8] K. Ulrich and D. Eppinger, "Product design and development," *McGraw-Hill/Irwin*, 2000.

[9] X. Fischer, *Stratégie de conduite du calcul pour l'aide à la décision en conception intégrée ; application aux appareils à pression [Ph.D. thesis]*, ENSAM, Paris, France, 2000.

[10] https://www.modelica.org/.

[11] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, San Diego, FL, USA, 1993.

[12] A. K. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, no. 1, pp. 99–118, 1977.

[13] U. Montanari, "Networks of constraints: Fundamental properties and applications to picture processing," *Information Sciences*, vol. 7, no. C, pp. 95–132, 1974.

[14] B. Yannou and A. Hamdi, "Truss dimensioning with an uncertainty reduction paradigm," in *Proceedings of the International Design Conference*, Dubrovnik, Croatia, 2004.

[15] L. Jaulin, M. Kieffer, and O. Didrit, *Applied Interval Analysis*, Springer, Berlin, Germany, 2001.

[16] L. Granvilliers and F. Benhamou, "Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques," *ACM Transactions on Mathematical Software*, vol. 32, no. 1, pp. 138–156, 2006.

[17] L. Michel and P. Van Hentenryck, "Helios: A modeling language for global optimization and its implementation in Newton," *Theoretical Computer Science*, vol. 173, no. 1, pp. 3–48, 1997.

[18] K. R. Apt and M. Wallace, *Constraint Logic Programming Using ECL* , Cambridge University Press, Cambridge, 2006.

[19] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím, "IBM ILOG CP optimizer for scheduling," *Constraints*, vol. 23, no. 2, pp. 210–250, 2018.

[20] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget, "Revising Hull and Box consistency," in *16th International Conference on Logic Programming*, pp. 230–244, MIT Press, Cambridge, MA, USA, 1999.

[21] J. C. Fauroux and P. Lafon, "Optimization of Multi-Stage Transmission Mechanism," in *Proceedings of the 11th World Congress in Mechanism and Machine Science IFToMM*, Tianjin, China, 2003.

[22] F. Benhamou and L. Granvilliers, "Chapter 16 Continuous and interval constraints," *Foundations of Artificial Intelligence*, vol. 2, no. C, pp. 571–603, 2006.

[23] G. B. Dantzig and M. N. Thapa, *Linear Programming 1: Introduction*, Springer-Verlag, New York, NY, USA, 1997.

[24] G. B. Dantzig and M. N. Thapa, *Linear Programming 2: Theory and Extensions*, Springer-Verlag, New York, NY, USA, 2003.

[25] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, NY, USA, 2006.

[26] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Oxford, UK, 1975.

[27] X. Yu, Y. Lu, and X. Yu, "Evaluating Multi objective Evolutionary algorithms using MCDM Method," *Mathematical Problems in Engineering*, vol. 2018, Article ID 9751783, 13 pages, 2018.

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[29] Federico Alonso-Pecina and David Romero, "A Simulated Annealing Approach for the Train Design Optimization Problem," *Mathematical Problems in Engineering*, vol. 2017, Article ID 4703106, 11 pages, 2017.

[30] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[31] Y. Deng, W. Zhu, J. Tang, and J. Qin, "Solving a Two-Stage Stochastic Capacitated Location-Allocation Problem with an Improved PSO in Emergency Logistics," *Mathematical Problems in Engineering*, vol. 2017, Article ID 6710929, 15 pages, 2017.

[32] L. Giraud and P. Lafon, "A comparison of evolutionary strategies for mechanical design," *Journal of Engineering Optimization*, vol. 34, no. 5, pp. 307–322, 2002.