INTERNATIONAL DESIGN CONFERENCE - DESIGN 2018
https://doi.org/10.21278/idc.2018.0499

SUPPORTING DESIGN TASKS THROUGH CONSTRAINT SATISFACTION TOOLS

M. Nardelli, P. Cicconi, R. Raffaeli and M. Germani

Abstract

Nowadays, different commercial tools are available to support engineers during optimization tasks in engineering design; however, many researches have been still studying tools and methods to improve the design process and overcome some limits related to configuration and design optimization. This paper proposes a methodological approach to highlight how a CSP analysis can support the first phase of an optimization analysis, to reduce the design space of solutions to be investigated and subsequently optimized. A test case shows a CSP study applied to steel structures for oil & gas applications.

Keywords: design optimisation, constraint modelling, engineering design, steel structures

1. Introduction

The design of mechanical products is often an integrated and collaborative process which regards the optimization of Constraints Satisfaction Problems (CSP). Many tools and research are available to support the optimization phase in mechanical design; however, a lack still exists in the development of a flexible and agile design process. Generally, designers use and manage design rules, tables, formulas, and relations during the mechanical design process. A CSP model implements all these knowledge representations as mathematical constraints. Trabelsi et al. (2015) applied a CSP method to support the preliminary design of a linear vehicle suspension system. This research, which was focused on the sizing of the system, proposes a comparison between conventional design methods and shows how the computation time related to CSP models is satisfactory.

Yvars (2011) introduced a deterministic CSP approach to solve a pareto bi-criterion optimization. They proposed a research example focused on a bolt coupling (Yvars, 2011). His research used CSP methods to reduce the size and the complexity of the design phase. However, this approach does not consider the integration of the design workflow with other tools such as external configuration software or numerical FEM solvers. The use of FEM solvers increases the calculation time and requires the use of other kinds of optimization algorithms. Yvars (2011) show how designer can test several ways by adding or deleting constraints in a CSP model. Raffaeli et al. (2017) also studied CSP problems. Their focus was on mechanical systems such as Engineer-To-Order (ETO) products. These complex products require a more complex formalization of knowledge. They studied a CSP approach for a further implementation into a generic configuration system. Generally, design parameters in CSP problems must satisfy a collection of constraints related to normatives, customer requirements, marketing requirements, etc. Some research proposes configuration approaches to solve CSP problems in different application fields. An ETO application was analysed by Cicconi et al. (2018). He proposes an approach to perform the configuration and the optimization of oil & gas ducts. However, his approach lacks a CSP module to reduce the variation of parameters to be optimized. Lin et al. (2017) proposed a TRIZ-based approach to support design when the solutions of optimization methods do not meet the objectives of problems to

solve. His approach is focused on the analysis of system contradictions which extracts data from the simulations generated from the optimization loop.
The CSP approach requires the definition of a mathematical model which reproduces the behaviour of the physical system (Guns et al., 2013). A software tool is necessary to apply this approach in mechanical design. The model to be applied is parametric; therefore, each variable can change is value in function of a defined variation range. Traditional CSP solvers iterate each configuration of parameters until the defined mathematical constraints are satisfied. However, in mechanical industry, many design problems allow numerous solutions to be applied. For this reason, a lot of research is focused on techniques and methods to support the design optimization. As described by Yvars (2011), the multi-objective optimization is an approach for satisfying the conflicting criteria. In the context of oil & gas applications, Cicconi et al. (2016) describe an optimization workflow to reduce the weight of big steel structures. This approach considers the integration of FEM simulations into the optimization analysis. Constraints were defined as normative checks to be verified after each structural calculation.
Regarding the use of configuration tools in mechanical design, the Rule Based Design (RBD) approach is quite used (Chavali et al., 2008) to implement tailored KBE (Knowledge Based Engineering) applications, using scripts and programming codes. The use of RBD programs enhances a reduction in design cycle time by 50% (Efthymiou et al., 2013). The current engineering design process shows an imbalance between the time required for non-creative activities and the time available for the exploration of innovative design spaces. This imbalance can become critical for complex products (La Rocca et al., 2012). A limit of traditional RBD systems is the difficult to solve open-ended problems such as optimization studies where the possibilities and design variations are difficult to predict.
Optimization analysis are based on searching the parameters configuration which optimizes the target objectives (Lin et al., 2017). In this approach, constraints are defined as limits, which are represented through Boolean checks (*if* <*condition*> *then* <*instruction-1*> *else* <*instruction-2*>), which are generally evaluated at the end of an optimization workflow. The condition to be verified is a rule or a formula which regards the know-how of the product knowledge. However, if an optimization analysis requires the employment of heavy time-consuming simulations, the verification of constraints is applied after the iteration of many simulations. Therefore, the optimization workflow can also generate many not-verified solutions. This case is not efficient because it employs computing resources to calculate different not-valid configurations. Additionally, an optimization analysis requires the definition of a range of values for each parameter to be analysed. This phase is manual, and it required the designer's know-how to configure each parameter range. Parametric template-models can help to develop a design automation workflow from configuration to optimization; however, the definition of the variation range is often an engineer's task and therefore a manual activity.
This paper proposes a CSP approach to overcome the two highlighted limits, which interest the phase of the design optimization. While the first limit regards the application of the constraints analysis only at the end of the design workflow, the second one is the difficulty to set the suitable parameter range to avoid the generation of not-satisfied solutions to be simulated. The aim of the research is to apply a CSP model to generate a space of satisfied solutions to be involved in a further optimization phase, to reduce the number of not-satisfied calculation before the simulation phase. The analysis of a CSP model can be defined as a first level of an optimization design process for reducing the parameter variations in further analysis. The CSP model can be solved using a combination of heuristics and combinatorial search methods to be solved in a reasonable time. Particularly, analytical models can be quickly solved; therefore, they can be employed in an optimization analysis.
The test case shows a CSP study applied to steel structures used in oil & gas applications. In this paper, a Gecode-based tool has been developed using MiniZinc programming language. The developed tool includes the mathematical formulation of design constraints used for the design of steel structures. In particular, the test case is focused on *Design Constraints*, without considering *Normative Constraints* or *Configuration Constraints* at this level.
The remainder of the paper introduces the state of the art about CSP problems and describes the proposed approach. A test case section shows an industrial application for the design of oil & gas steel constructions, called as "module". In the context of mechanical design, the results show how a Gecode-

based tool can be used as a customized solver for CSP models to reduce the space of solutions to be analysed in an optimization phase.

2. State of the art

Computational Design Synthesis (CDS) is a research area focused on activities to automate the design phase in the production. CDS must be able to generate automatically alternatives and to avoid long and useless routine works (Campbell and Shea, 2014). Some tools have been developed but they remain too time and memory consuming. Mueller et al. (2015) applied this method to design and optimize parts for additive manufacturing processes. In his work he analyzed parameters of the 3D printing process and gathered the necessary information to understand variations of 3D printed structures. The resulting parameters led to build accurate part models and optimize, fabricate, and test them in the best way.

CDS is used in every Knowledge Based Engineering (KBE) application to automate design routines and to implement a multidisciplinary product design. More specifically KBE tries to increase complex and iterative calculation with geometrical modelling. KBE is in fact a research field that studies methodologies and technologies for capturing and re-using product and process engineering knowledge to achieve automation of repetitive design (Verhagen et al., 2012).

Therefore, a KBE approach also concerns generative geometry, ability to write reports, ability to increase automation and extend features in CADs tools, reuse of design rules and engineering knowledge (La Rocca et al., 2012). This technology has its roots in Artificial Intelligence (AI) and in Knowledge Base Systems (KBS). La Rocca describes a typical KBE application for aircraft applications. This is an applicative case that shows how this technique is efficient in helping the designer to analyze and solve problems that must face frequently (Zhu et al., 2017).

One of the hard challenges in the aircraft design is the modelling of the harness 3D routing. The difficulty stays in its intrinsic complexity, but also in the increasing of the design constraints and in its dependency on any little change. Zhu et al. (2017) proposed methods based on KBE and optimization to reach minimum cost routing solutions that satisfy constraints of all relevant design rules.

In all cases, despite how they are called, during the design phase the designer has to face, consciously or not, a set of rules to solve. This set of rules in Mathematics is named Constraint Satisfaction Problem (CSP). They could be defined as objects that must satisfy a set of constraints. These problems are tasks for research in AI and operational research. They are extremely interesting; in particular, CDS is a powerful mean to express engineering problems and pursuit solutions in product design supporting the knowledge-intensive process (Helms et al., 2009).

Many design problems are analyzed in literature using a CSP approach with design synthesis. An example is in the field of robotics systems (Trabelsi et al., 2015), where Chen developed a simple design principle for an untethered, entirely soft, swimming robot with the ability to achieve directional propulsion without batteries and on-board electronics. A multidisciplinary design optimization advisor system (Sobieszczanski-Sobieski et al., 2015) and the implementation of an ICAD generative model (La Rocca et al., 2002) was proposed in the design of aircraft applications.

There are numerous methods to solve this kind of problems and many other methods are also studied to improve calculation time and precision in CSP problems. The more common algorithms are: *backtracking*, *branch and bound*, and *depth first search*. They represent three categories of methods to solve CSP problems. Many variations and improving of them have been studied and implemented in literature.

Backtracking is a technique that consists to enumerate all possible solutions and prunes all the ones that do not satisfy some constraint (Kumar and Lin, 1986). It explores a graph tree remembering all the nodes already analyzed, in that way, if a path must be pruned, it can come back to the node that is not visited jet without the risk to move in a path already explored. The *backtracking* algorithm has an exponential complexity, so it is not so efficient for not NP-complete problems. Nevertheless, the algorithm integrates some heuristic techniques that allow to decrease complexity.

Branch and bound is a general technique to solve finite optimization problems (Morrison et al., 2016). This approach starts subdividing the original problem in subproblems, which are easier to solve. *Branch and bound* algorithms are called as implicit enumeration because they enumerate all the possible solutions and tries all of them, but some will be deleted proving their non-optimality.

Depth first search is another technique of tree search, its particularity is that the algorithm can explore nodes far from the root without having visited the nodes of first generation (Mehlhorn and Sanders, 2008). The search strategy explores the graph reaching the deepest possible node in it. Once all the deepest nodes are analyzed, the algorithm comes back to explore all the previous ones.
A development platform that implements all these algorithms and many others is Gecode, which is an open, free and fast toolkit used to solve CSP problems. Gecode already contains many features but its strength is in the fact that is programmable: it also supports the implementation of new constraints, strategies, and search methods (Schulte and Tack, 2006). The programming language for extensions is C++ based. Moreover, Gecode is efficient in time and memory consuming. It has been used to solve over 50000 test cases with good results. However, Gecode has some limitations: it does not support geometrical constraints, it does not support connections with database, it has also limitations using constraints that involves strings, finally the connection with other software must be implemented with custom libraries.
Münzer (2015) used Gecode to implement part of a model to search design solutions derived from a set of specified techniques. This method is a concrete application of the CDS. The method consists in: define a metamodel of the problem and its constraints, find the interconnections, assign variables using CSP problems (Gecode) and optimize an objective function using simulating annealing.
Other tools have been analyzed to solve CSP problems such as MiniZinc (Nethercote et al., 2007), FlatZinc, and Choco. MiniZinc is a medium-level constraint modelling platform based on Zinc language (Guns et al., 2013). It can be mapped onto existing solvers. Algorithms developed with MiniZinc are compiled in FlatZinc language. FlatZinc is a low-level programming language which can directly interact with solver such as Gecode and others (Urli et al., 2015). Therefore, FlatZinc is used to translate the CSP model into the language required by the defined solver.
Choco is another free open-source Java library dedicated to CSP problems (Prud'homme, 2017). The user can define its problem in a declarative way by stating the constraints that he wants to satisfy. Then, the related CSP model is solved by alternating constraint filtering algorithms with a search engine.
These tools and others have all the same problem that they cannot be programmed; thus, plug-in and extensions are not allowed. Considering these limitations, the proposed paper uses Gecode with MiniZinc as solution language to model and solve CSP problems.

3. Approach

3.1. Method

Figure 1 shows the overall methodological approach proposed in this paper. The scope of the research is the definition of tools and methods to supply the design phases from configuration to optimization. The aim regards the reduction of time in mechanical design using feasible and automatic design tools. These tools implement knowledge base formalized in rules, formulas, and constraints. In fact, a part of this research is focused on the implementation of CSP models and the development of programming codes to describe typical constraints used in mechanical design. While next section deals with the approach used to develop the software tools, including algorithms to customize the modelling of mathematical constraints, this section describes the proposed design approach, where design constraints are implemented in a CSP model.
Three constraints levels are highlighted in Figure 1: *Configuration Constraints*, *Design Constraints*, and *Performance Constraints*. This classification aims to investigate the different types of constraints inside a typical mechanical design, which is based on configuration and optimization phases. In particular, *Configuration Constraints* are those validation rules and formulas which mainly are related to the implicit knowledge of the engineer. An example could be the compatibility analysis between two options; thus, this level introduces constraints to evaluate the product structure of a configuration. The level called *Performance Constraints* is focused on normative checks and conditions such as structural limits to be applied; therefore, this level mainly concerns the explicit knowledge. On the other hand, the *Design Constraints* level highlights validation rules mainly focused on the domain of the technical know-how. These constraints can refer to the formalization of explicit and implicit knowledge. Examples can be validation rules to check the weight balance of assemblies into a steel structure.

Another example can be rules to evaluate and limit the number of different types of components, such as the types of beams into an assembly.

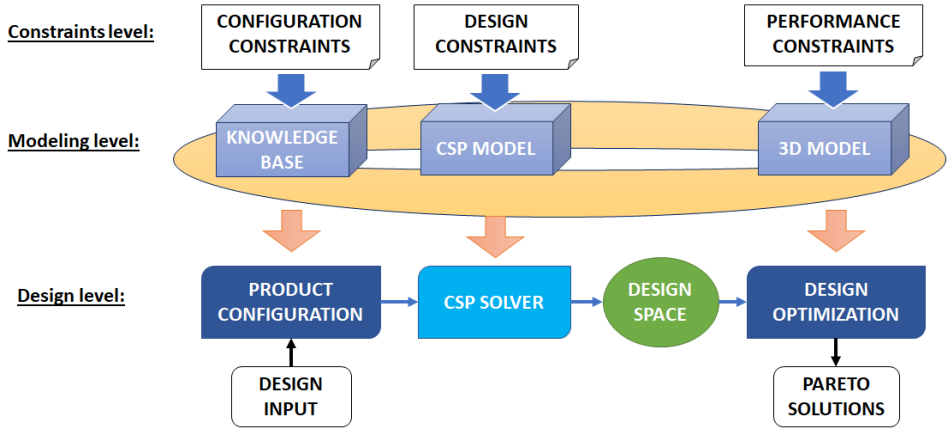


Figure 1. The overall methodological approach

Summarizing, if *Performance* and *Configuration Constraints* are mainly related to normative and setup rules, *Design Constraints* are focused on practical satisfaction rules which depends on the engineer's know-how. In this context, *Configuration Constraints* are usually implemented in configuration tools such as filtering and validation rules, and *Performance Constraints* are defined as check conditions into optimization loops after simulation analysis. The collection of *Design Constraints* is often defined into previous two levels; however, the proposed approach highlights the necessity to use an intermediate level. The CSP model, where *Design Constraints* are applied, is a rapid rule solver which can evaluate many parametric configurations in few seconds, without interfacing with external computing such as FEM analysis. In fact, a classical CSP tool is only based on mathematical equations and disequations. This approach can reduce the number of solutions to be evaluated in a next and more complex optimization loop, which involves a time-consuming analysis. This intermediate level is also suitable to avoid technical rules inside a configuration workflow; in order to separate knowledge focused on a specific configuration's workflow from technical knowledge which could be generic and valid for different cases.

The use of a CSP model as proposed (Figure 1) can also perform the automatization from configuration to optimization. In fact, the outcome of a configuration process is often a parametric model, which can be optimized in a further step of analysis. However, it is often difficult to define a variation range for each variable. In this case, the solution of a CSP model can evaluate a design space of solutions to reduce the number of combination which don't satisfy some design constraints. Therefore, this approach allows different not-satisfied configuration to be analyzed with a CSP model and to be excluded from simulations and optimization loop.

3.2. Development approach

The method proposes a software application which is based on Gecode toolkit. In this approach, while Gecode runs in the background, an interface shows and manages the CSP problem. Figure 2 describes the methodological workflow from the CSP definition to results.

The first step is the definition of the CSP problem. This means to translate something that often is only in the mind of the designer in a standard understandable problem, made of variables and constraints. All the technical and geometrical specifications must be rewritten in a universal language that anyone can read. Therefore, the approach considers the implementation of a formalism to provide the input definition. The user can add constrains compliant to the defined formalism. The formalization of the technical knowledge, which is required to define constraints in the CSP definition, is a difficult phase because design routines are often based on experience. They are not related to standard procedures. While explicit knowledge is easy to translate in a mathematical constraint, implicit and tacit knowledge are very difficult to be elicited and formalized.

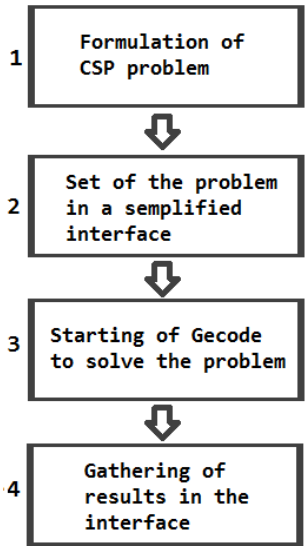


Figure 2. The software development workflow

The second step is to write the CSP problem in a simplified Graphical User Interface (GUI), which uses Gecode to solve the CSP problem. Figure 7 describes the proposed GUI, which consists of four levels: *Variables*, *Constraints*, *Solver Method*, and *Results*. The first level concerns the settings and management of variables using a table. The *Constraints* level regards the definition of the mathematical rules to be verified in the CSP calculation. While the solver method to be applied is defined in the third level, the forth level shows the results. After pushing the Solve button (Solver Method level in Figure 7), Gecode is invoked. The CSP definition is translated in Gecode language for the computing. In fact, the proposed application uses Gecode algorithms to solve CSP problems. Gecode elaborates the instructions to be solved, and then returns results.

4. Case study

The proposed test case proposes the CSP modelling of a big steel structure called module, which is a big 500-ton steel construction used for power applications in oil & gas. Firstly, this section introduces the context related to oil & gas modules. Secondly, the CSP model is described with parameters and constraints. The CSP study focuses on some constraints related to the structure's weight and frame sizing. In particular, groups with pipe frame were analysed in the proposed CSP module. The objective of the test case is to reduce the space of all possible solutions to be optimized after a first CSP analysis. Despite the verticalization of this case study, this CSP tool remains a general tool. In fact, any product structure can be added and modified to describe a general CSP model with relative constraints. As said before, the difficult part is to formalize these constraints in a mathematical way. After this step, the implementation of the problem is quite intuitive using the proposed interface.

4.1. Oil & gas module

An oil & gas plant is often a collection of pre-fabricated modules. In fact, modularization and pre-assembly are common design strategy in oil & gas plants. In particular, a module is the smallest functional unit with its equipment, machines and steel structure. Therefore, a single module (Figure 3) can contain power generation units, gas compression units, and process equipment for oil & gas applications. The modularity approach for oil & gas plants reduces the overall cost and the delivery time. The structure of a classic module consists of steel beams used for internal support of machines and equipment. The structure can be divided in the supporting frame (or main frame) and the secondary one. The secondary level of the structure regards the supports for minor equipment. In general, the secondary frames and braces increase the stiffness of the structure of the module. The oil & gas module, proposed in this test case, is an example of a 500-ton construction with three levels. Figure 4 shows the relative FEM model defined using the SAP2000 software.

Figure 3. An example of an oil & gas module during the installation phase

A list of Design Constraints has been analysed and discussed with an expert team of designers. As example, one of the analysed design constrains regards the sizing of frames. The ratio between column's section and height is a limit constraint for checking the buckling analysis. This limit can be evaluated using analytical formulas which depends on the geometry of the beam section. The buckling check is a typical design constraint which can be evaluated with an analytical approach before to run a FEM (Finite Element Methods) analysis. Another constraint is related to the sizing of pipe beams. The ratio diameter per width is a constraint which can be described using a mathematical formulation because the limit value depends on the diameter range. Constraints can be also applied to verify the dimensions of each beam at a connection node. Additional constraints have been defined on the construction's weight. Constraints can be applied to regulate the weight balance of each group of beams. The steel structure's weight can be also considered as a constraint for the CSP analysis and for reducing the number of combinations to be evaluated in a further optimization phase.

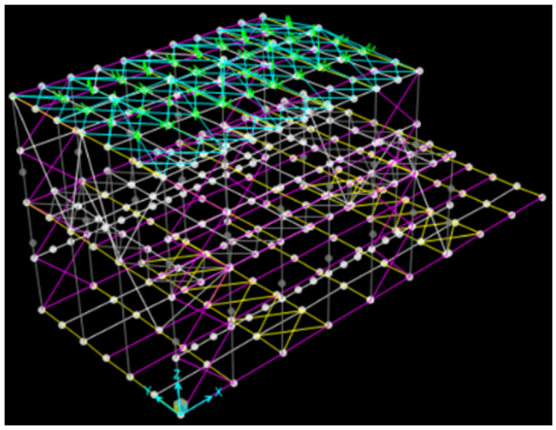


Figure 4. The structure modelling used for the FEM analysis

4.2. Implementation and results

The product structure of the proposed *module* and the frame collection have been acquired from SAP2000, which is the FEM tool employed in this test case. An import tool has been developed using VB.NET and implementing API (Application Programming Interface) tools provided by SAP2000. The imported structure has been formalized in an UML class diagram.

As explained in the beginning of the section, the test case proposes the CSP modelling of a steel structure. In particular, this structure, called *module*, consists of groups; each group is characterized for its type and its function. Types can be: standard, pipe, or built-up.

The principal task is to find the minimum weight configuration varying external diameters and thickness of the pipes of the structure. The constraints are the dimensional tolerances for the diameter and thickness and the range of the ratio between the diameter and the thickness must respect certain limits.

Some support variables and support constraints are added for the weights of the groups.
Figure 5 shows the developing phases, which regards the employment of different programming tools such as VB.NET, MiniZinc, and scripts. Firstly, the authors have been implemented all the classes necessary to manage the model; these are involved in three projects, developed in VB.NET. One is entirely dedicated to the interface, here there are implemented all the forms that allow the user to add variables and constraints. The scope is to ease the user to define the model without a complication due to hard programming languages and for reduce time consuming. Another project is the generalization of the model. One class represents the model that is made of variables, constants, and constraints. Variables and constraints are also classes. Another class is implemented to manage the model. The last project is the specialization of the model for this case study. It represents a beams module. The classes needed are the types of the beams and the manager of the module. In these classes there all the functions to write a script to send to MiniZinc, the program that can solve CSP problems. After MiniZinc has analysed and solved the problem, the results are gathered in the interface and shown to the user.

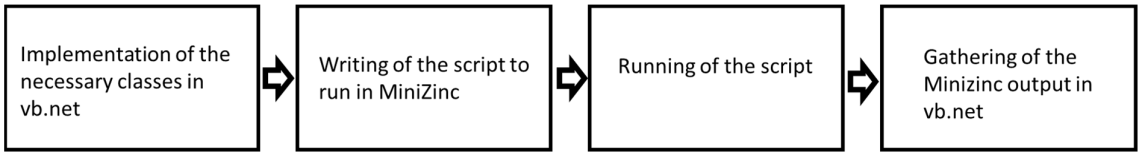
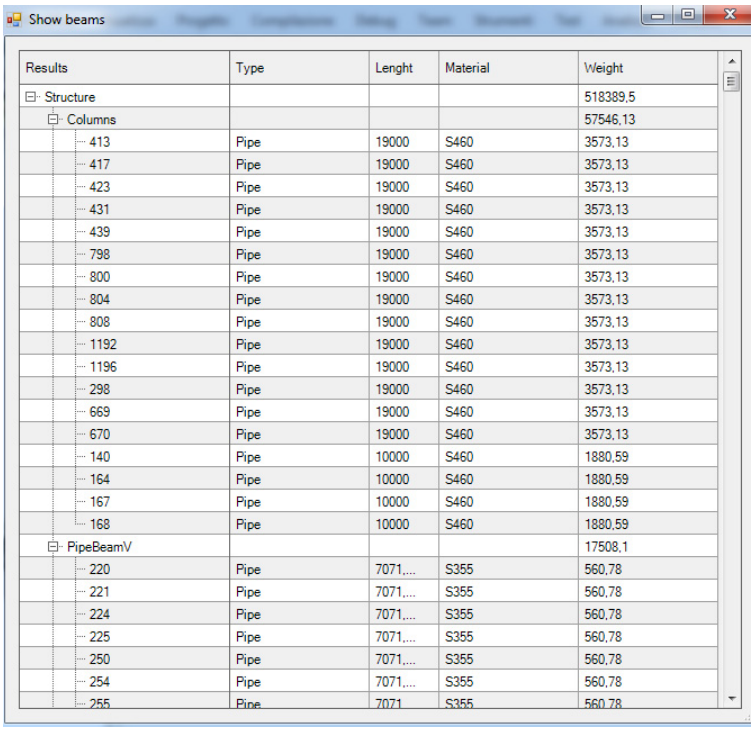


Figure 5. Developing phases from VB.NET to MiniZinc

All the constraints are implemented in a VB.NET application, which fills the problem simply reading the xml file with the structure of the steel *module* (Figure 6). A function, called *Fill Beams*, reads data and automatically gathers all the information necessary to solve a minimization problem for the weight of the structure.



Results	Type	Lenght	Material	Weight
Structure				518389,5
Columns				57546,13
413	Pipe	19000	S460	3573,13
417	Pipe	19000	S460	3573,13
423	Pipe	19000	S460	3573,13
431	Pipe	19000	S460	3573,13
439	Pipe	19000	S460	3573,13
798	Pipe	19000	S460	3573,13
800	Pipe	19000	S460	3573,13
804	Pipe	19000	S460	3573,13
808	Pipe	19000	S460	3573,13
1192	Pipe	19000	S460	3573,13
1196	Pipe	19000	S460	3573,13
298	Pipe	19000	S460	3573,13
669	Pipe	19000	S460	3573,13
670	Pipe	19000	S460	3573,13
140	Pipe	10000	S460	1880,59
164	Pipe	10000	S460	1880,59
167	Pipe	10000	S460	1880,59
168	Pipe	10000	S460	1880,59
PipeBeamV				17508,1
220	Pipe	7071,...	S355	560,78
221	Pipe	7071,...	S355	560,78
224	Pipe	7071,...	S355	560,78
225	Pipe	7071,...	S355	560,78
250	Pipe	7071,...	S355	560,78
254	Pipe	7071,...	S355	560,78
255	Pipe	7071	S355	560,78

Figure 6. Module's structure imported from SAP2000 and exported in xml format (lengths are expressed in meters and weights in kg)

The computational model of a CSP problem is divided in variables, constants, and constraints. Figure 7 shows constraints definitions and ranges of variables. The *Solve* command runs a simplified optimization using MiniZinc. This run is based on the definition of the defined constraints.

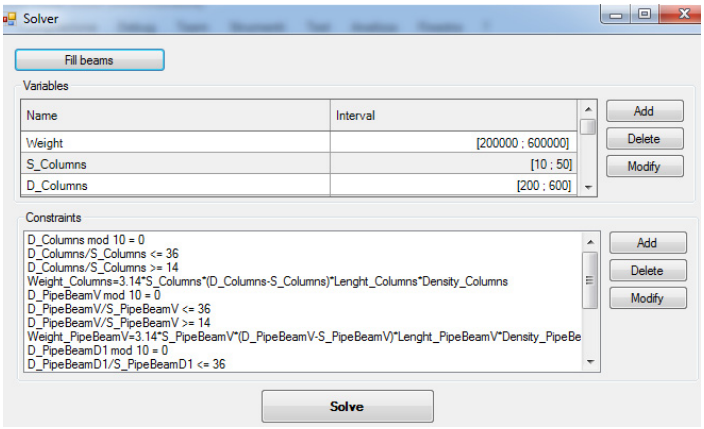


Figure 7. Settings interface and first optimization results

4.2.1. MiniZinc settings

This sub-section describes the information which is sent to MiniZinc. This process runs in the background; therefore, the code is hide to the user. This code, developed in VB.NET, starts the MiniZinc process without using a user graphical interface. Figure 8 shows the constants of the CSP problem. Groups with fixed weight, that is because the beams are of standard type, have only one constant: the weight. Groups with variable weight, pipe beam, have three constants for each group: the total length of the group, the count of the group and the density of the group material.

The variables are the external diameter and thickness. These two variables are replicated for each group of pipe beam involved. As shown in Figure 9 only four groups are made of pipes. For each group there is also the variable "Weight_<GroupName>" that is a support variable. Because the task is to minimize the weight also the total weight becomes a variable ("Weight").

```
float: Lenght_Columns= 306000;
float: GroupCount_Columns= 18;
float: Density_Columns= 7.73E-06;
float: Lenght_PipeBeamV= 220766.077366437;
float: GroupCount_PipeBeamV= 32;
float: Density_PipeBeamV= 8.06E-06;
float: Weight_BeamD0MBT500= 42636.2384695311;
float: Weight_BeamD0MBL400= 12186.72;
float: Weight_BeamD1MBT400= 60573.5232694227;
float: Lenght_PipeBeamD1= 648822.910957727;
float: GroupCount_PipeBeamD1= 142;
float: Density_PipeBeamD1= 8.06E-06;
float: Weight_BeamD0SB300= 14731.1494530692;
float: Weight_CrossBeam= 88284.0816;
float: Weight_CrossBeamDeck= 25996.8852;
```

Figure 8. Example of constants declaration

```
var float: Weight;
var 10..50: S_Columns;
var 200..600: D_Columns;
var float: Weight_Columns;
var 10..50: S_PipeBeamV;
var 200..600: D_PipeBeamV;
var float: Weight_PipeBeamV;
var 10..50: S_PipeBeamD1;
var 200..600: D_PipeBeamD1;
var float: Weight_PipeBeamD1;
var 10..50: S_PipeBeamD2;
var 200..600: D_PipeBeamD2;
var float: Weight_PipeBeamD2;
```

Figure 9. Example of variables declaration

Figure 10 shows the declaration of constraints in MiniZinc language. They consist of geometrical limits and support constraints.

```
constraint D_Columns mod 10 = 0;
constraint D_Columns/S_Columns <= 36;
constraint D_Columns/S_Columns >= 14;
constraint Weight_Columns=3.14*S_Columns*(D_Columns-S_Columns)*Lenght_Columns*Density_Columns;
constraint D_PipeBeamV mod 10 = 0;
constraint D_PipeBeamV/S_PipeBeamV <= 36;
constraint D_PipeBeamV/S_PipeBeamV >= 14;
constraint Weight_PipeBeamV=3.14*S_PipeBeamV*(D_PipeBeamV-S_PipeBeamV)*Lenght_PipeBeamV*Density_PipeBeamV;
constraint D_PipeBeamD1 mod 10 = 0;
constraint D_PipeBeamD1/S_PipeBeamD1 <= 36;
constraint D_PipeBeamD1/S_PipeBeamD1 >= 14;
constraint Weight_PipeBeamD1=3.14*S_PipeBeamD1*(D_PipeBeamD1-S_PipeBeamD1)*Lenght_PipeBeamD1*Density_PipeBeamD1;
constraint D_PipeBeamD2 mod 10 = 0;
```

Figure 10. Example of constraints declaration

4.2.2. Calculation and results

An additional collection of constraints has been added. The expert designer defined rules to limit the weight related to some groups of beams. Figure 11 describes a list of constraints which regulate the weight balance of a module's structure. Certain parts of the structure must respect a percentage weight range against to the total weight.

```
constraint Weight_Columns<=0.15*Weight;
constraint Weight_Columns>=0.1*Weight;
constraint Weight_BeamD0MBT500+Weight_BeamD0MBL400+Weight_BeamD0SB300<=0.18*Weight;
constraint Weight_BeamD0MBT500+Weight_BeamD0MBL400+Weight_BeamD0SB300>=0.1*Weight;
constraint Weight_PipeBeamD1+Weight_BeamD1MBT400<=0.18*Weight;
constraint Weight_PipeBeamD1+Weight_BeamD1MBT400>=0.15*Weight;
constraint Weight_PipeBeamD2+Weight_BeamD2MBL500+Weight_BeamD2MBT500<=0.15*Weight;
constraint Weight_PipeBeamD2+Weight_BeamD2MBL500+Weight_BeamD2MBT500>=0.1*Weight;
```

Figure 11. Example of additional constraints used for the weight balance

The proposed calculation regards a searching of all solutions which satisfy the defined constraints. A programming script runs the solution using MiniZinc, which is a tool based on Gecode platform. Figure 12 describes the graphical report of the CSP solution. The solutions which satisfy each constraint are 74 against an initial number of about 2000 combinations. Admissible solutions are highlighted in green and yellow. The time to compute the problem is 1s 572msec.

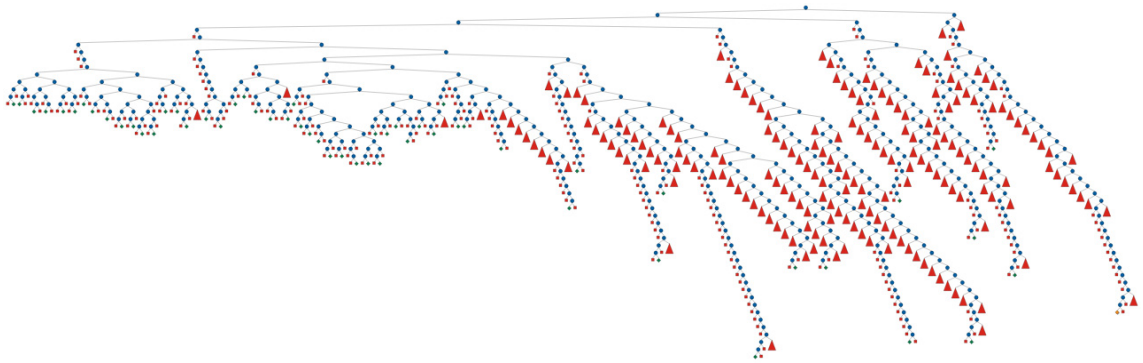


Figure 12. A graphical report of the CSP solution

The achieved CSP solutions show a weight range between 489 ton and 625 ton. The configuration of the minimized solution is highlighted in Figure 13. The 8 parameters describe diameter and thickness related to four groups of tubular beams. While the weight value is expressed in kg (Figure 13), the diameter and thickness values are evaluated in mm.

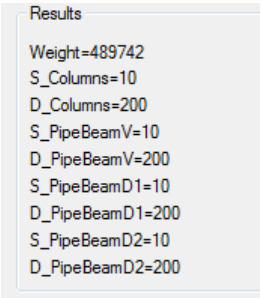


Figure 13. The result of the CSP optimization which minimizes the *module*'s weight for the 8 parameters analysed

As a future development, the resultant list of the CSP solution can be used as a basis for a further genetic algorithm optimization. The second step of the design optimization will introduce FEM tools for the detailed calculation of the mechanical behaviour.

Using the proposes CSP tool, the test case shows an important time reduction of about 7 working days for the conceptual design phase. The resulting lead time for the structural optimization is about 3 days against an average period of 10 working days. Additionally, this reduction allows the time related to the proposal submission to be reduced. Generally, a proposal submission takes from 4 to 8 weeks. A reduction time of about 1 week can be suitable for the successful closure of the proposal.

5. Conclusions

A methodological approach has been proposed to highlight how a CSP analysis can support the first phase of an optimization analysis, to reduce the design space of solutions to be investigated and subsequently optimized. The analysed CSP model is a connecting point between configuration and design optimization. The calculation velocity of CSP problems (about 1.5s) can be used to develop automatic procedures for improving the integration between configurators and optimization tools. The optimization analysis, based on simulations, also requires the necessity to set constraints to verify the product performance. Even if different constraints, related to simulation results, must be always analysed after an optimization loop, some constraints can be analysed using analytical model before the run of an optimization loop.

The paper shows how a CSP approach can be used to obtain a first level of satisfied solutions solving analytical design constraints without running heavy-computing simulations such as FEM analysis. The analysed results highlight 74 solutions from a set of about 2000 combinations.

The article has been verticalize for this case study; however, the interface permits to add different types of models and constraints. The proposed CSP tool is a generic framework to build a product structure using an O-O programming. Therefore, a designer can generalize this tool implementing a specific CSP model with its relative constraints.

As a future development, the second step of the design optimization will consider FEM tools for the detailed calculation of the mechanical behaviour. However, the further optimization will be based on the reduced list of solutions elaborated from the CSP analysis. Another future improvement could be the development of extensions features such as propagators of constraints to support mechanical design.

References

Campbell, M.I. and Shea, K. (2014), "Computational design synthesis", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 28 No. 3, pp. 207-208. https://doi.org/10.1017/S0890060414000171

Chavali, S.R., Sen, C., Mocko, G.M. and Summers, J.D. (2008), "Using Rule Based Design in Engineer to Order Industry: An SME Case Study", *Computer-Aided Design and Applications*, Vol. 5 No. 1-4, pp. 178-193. https://doi.org/10.3722/cadaps.2008.178-193

Cicconi, P., Germani, M., Bondi, S., Zuliani, A. and Cagnacci, E. (2016), "A Design Methodology to Support the Optimization of Steel Structures", *Procedia CIRP*, Vol. 50, pp. 58-64. https://doi.org/10.1016/j.procir.2016.05.030

Cicconi, P., Raffaeli, R., Marchionne, M. and Germani, M. (2018), "A Model-Based Simulation Approach to Support the Product Configuration and Optimization of Gas Turbine Ducts", *Computer-Aided Design and Applications*, Vol. 15 No. 6.

Efthymiou, K., Sipsas, K., Mourtzis, D. and Chryssolouris, G. (2013), "On an Integrated Knowledge based Framework for Manufacturing Systems Early Design Phase", *Procedia CIRP*, Vol. 9, pp. 121-126. https://doi.org/10.1016/j.procir.2013.06.179

Guns, T., Dries, A., Tack, G., Nijssen, S. and De Raedt, L. (2013), "The MiningZinc Framework for Constraint-Based Itemset Mining", *2013 IEEE 13th International Conference on Data Mining Workshops*. https://doi.org/10.1109/ICDMW.2013.38

Helms, B., Shea, K. and Hoisl, F. (2009), "A Framework for Computational Design Synthesis Based on Graph-Grammars and Function-Behavior-Structure", *14th Design for Manufacturing and the Life Cycle Conference; 6th Symposium on International Design and Design Education; 21st International Conference on Design Theory and Methodology*. https://doi.org/10.1115/DETC2009-86851

Kumar, V. and Lin, Y.-J. (1986), "A framework for intelligent backtracking in logic programs", *Foundations of Software Technology and Theoretical Computer Science, International Conference on Foundations of Software Technology and Theoretical Computer Science*, pp. 108-123. https://doi.org/10.1007/3-540-17179-7_7

La Rocca, G. (2012), "Knowledge based engineering: Between AI and CAD, Review of a language based technology to support engineering design", *Advanced Engineering Informatics*, Vol. 26 No. 2, pp. 159-179. https://doi.org/10.1016/j.aei.2012.02.002

La Rocca, G., Krakers, L. and van Tooren, M. (2002), "Development of an ICAD Generative Model for Blended Wing-Body Aircraft Design", *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. https://doi.org/10.2514/6.2002-5447

Lin, L., Rasovska, I., De Guio, R. and Dubois, S. (2017), "Optimization Methods for Inventive Design", In: Cavallucci, D. (Ed.), *TRIZ – The Theory of Inventive Problem Solving*, pp. 151-185. https://doi.org/10.1007/978-3-319-56593-4_7

Mehlhorn, K. and Sanders, P. (2008), *Algorithms and Data Structures: The Basic Toolbox*, Springer, Berlin. https://doi.org/10.1007/978-3-540-77978-0

Menouer, T. and Cun, B.L. (2013), "A Parallelization Mixing OR-Tools/Gecode Solvers on Top of the Bobpp Framework", *Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. https://doi.org/10.1109/3PGCIC.2013.42

Morrison, D.R., Jacobson, S.H., Saupee, J.J. and Sewell, E.C. (2016), "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning", *Discrete Optimization*, Vol. 19, pp. 79-102. https://doi.org/10.1016/j.disopt.2016.01.005

Mueller, J., Shea, K. and Daraio, C. (2015), "Mechanical properties of parts fabricated with inkjet 3D printing through efficient experimental design", *Materials & Design*, Vol. 86, pp. 902-912. https://doi.org/10.1016/j.matdes.2015.07.129

Münzer, C. (2015), Constraint-Based Methods for Automated Computational Design Synthesis of Solution Spaces, PhD thesis, Technische Universität München.

Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J. and Tack, G. (2007), "MiniZinc: Towards a Standard CP Modelling Language", In: Bessière, C. (Ed.), *Principles and Practice of Constraint Programming – CP 2007. CP 2007, Lecture Notes in Computer Science, Vol. 4741*, Springer, Berlin, pp. 529-543. https://doi.org/10.1007/978-3-540-74970-7_38

Prud'homme, C. (2017), *Choco-tuto Documentation*. [online] Available at: https://media.readthedocs.org/pdf/choco-tuto/latest/choco-tuto.pdf (accessed 11.12.2017).

Raffaeli, R., Savoretti, A. and Germani, M. (2017), "Design knowledge formalization to shorten the time to generate offers for Engineer To Order products", In: Eynard, B., Nigrelli, V., Oliveri, S., Peris-Fajarnes, G. and Rizzuti, S. (Eds.), *Advances on Mechanics, Design Engineering and Manufacturing, Lecture Notes in Mechanical Engineering*, Springer, Cham, pp. 1107-1114. https://doi.org/10.1007/978-3-319-45781-9_110

Schulte, C. and Tack, G. (2006), "Views and Iterators for Generic Constraint Implementations", In: Hnich, B., Carlsson, M., Fages, F., Rossi, F. (Eds.), *Recent Advances in Constraints. CSCLP 2005. Lecture Notes in Computer Science, Vol. 3978*, Springer, Berlin, Heidelberg, pp. 118-132. https://doi.org/10.1007/11754602_9

Sobieszczanski-Sobieski, J., Morris, A. and Van Tooren, M. (2015), *Multidisciplinary design optimization supported by knowledge based engineering*, Wiley. https://doi.org/10.1002/9781118897072

Trabelsi, H., Yvars, P.-A., Louati, J. and Haddar, M. (2015), "Interval computation and constraint propagation for the optimal design of a compression spring for a linear vehicle suspension system", *Mechanism and Machine Theory*, Vol. 84, pp. 67-89. https://doi.org/10.1016/j.mechmachtheory.2014.09.013

Urli, T., Ceschia, S., Schaerf, A. and Di Gaspero, L. (2015), "A General Local Search Solver for FlatZinc", *Metaheuristics International Conference, Marocco*.

Verhagen, W.J.C., Bermell-Garcia, P., van Dijk, R.E.C. and Curran, R. (2012), "A critical review of Knowledge-Based Engineering: An identification of research challenges", *Advanced Engineering Informatics*, Vol. 26 No. 1, pp. 5-15. https://doi.org/10.1016/j.aei.2011.06.004

Yvars, P.-A. (2011), "Pareto Bi-Criterion Optimization For System Sizing: A Deterministic And Constraint Based Approach", *International Conference On Engineering Design, ICED11*.

Zhu, Z., La Rocca, G. and van Tooren, M.J.L. (2017), "A methodology to enable automatic 3D routing of aircraft Electrical Wiring Interconnection System", *CEAS Aeronautical Journal*, Vol. 8 No. 2, pp. 287-302. https://doi.org/10.1007/s13272-017-0238-3

Dr. Paolo Cicconi
Università Politecnica delle Marche, DIISM
via Brecce Bianche 12, 60131 Ancona, Italy
Email: p.cicconi@staff.univpm.it