Computation of Ranges in Interval-based Constraint-Geometry of Building Models

Kirchner, J.
Fachgebiet Bauinformatik, TU Berlin, Germany
jakob.kirchner@tu-berlin.de

Abstract. Determining the optimal design of geometry in building models is an essential task for architects and engineers. Although many constraints are considered, only a small part is currently integrated into digital models as design knowledge. Current software products do not support the declarative approach of creating constraints and permissible ranges to parameter values. The reason is the algorithmic complexity in analyzation and evaluation of such a constraint model. In this paper the problem is summarized and a solution for the computation of possible ranges is presented. It makes use of existing algorithms. This research is part of the so-called geometric constraint satisfaction problem (GSCP). Although former approaches exist, they weren't applicable to the general case. The success of this approach is shown for a selected example. Future steps are discussed. These would be required for a modelling software benefitting from this approach.

1. Introduction

The state of the art in the creation of digital building models including 3D geometry requires an iterative process. This includes the placement of building objects and checking whether the current model meets certain requirements. These requirements comprise rules from standards and laws, but also originate from project contracts, architectural design intent, feasibility of building processes or valid geometric consistency. The paradigm of parametric modeling is common for adjusting a model to certain requirements, creating variants or reducing the time needed for subsequent modifications. Therefore, input parameters and computation rules are used to create a computational system. This can be used for an efficient recomputiation of a solution whenever a parameter value gets changed. One approach is the modeling of dependencies between the parameters as a computational plan. This concept distinguishes between input and output parameters and can be represented as a directed acyclic graph with edges describing the computational direction. A different approach is the use of so-called constraints which includes dependencies between parameters with no predefined computational direction. The approach is declarative and usually requires a solver with sophisticated algorithms to obtain solutions. That's because, a computational plan has to be identified or an equation system has to be solved. In complex models including many dependencies the computational plan and the feasible values for the parameters cannot be obtained easily because of the structural properties of the modeled problem.

Current modeling software like Autodesk Revit 2022, AutoCAD 2022, ArchiCAD 25 by Graphisoft is based on the principle of parametric design, but does only support single values for each parameter value. The basic principle - although not explicitly mentioned by the companies and not every piece of information is presented to the user – is apparently history-based design and its variants (Shah, 2001). History-based design requires a strict modeling direction. Although variational design by constraints (Shah, 2001) is also supported in these software products, it is limited to 2-dimensional sketches as parts of the construction history tree. In practice these limits prevent a massive use of constraints regarding the optimization of design. Nevertheless, there exist practical approaches for optimizing the design, e.g. generative methods or machine learning. As a limitation, these approaches allow only as many constraints

https://doi.org/10.7146/aul.455.c211

as needed for ensuring a single solution can be computed. Additional constraints cannot be considered or have to be validated iteratively. Alternative visions describe a software environment which can be used to control the number of variants by adding, removing or modifying requirements. Variants would be ready for examination and for adjusting the limits (Bettig et al. 2005, p. 8). Valdes et al. (2016) expect such approaches to increase the consistence of a model and the reduction of errors.

Another limit in current CAD and modeling software is the support for exactly one existing value for each parameter or coordinates. This principle known as *point-based design* leads to the situation where the possible ranges of parameter values are not clearly evident. In contrast to that, the idea of set-based design was introduced by the Japanese automobile company Toyota (Sobek et al. 1999) and has the goal of representing all sections of the design space. The design is refined by narrowing and cutting the remaining design space until the project converges to a point-based state (Liker et al. 1996, p. 167). Applying the idea of set-based design to digital models was already proposed for building projects (Gil et al. 2008). Yannou et al. (2013) developed a small model of interval values for parameters and computed possible values using interval arithmetic. Assuming a working software product supporting set-based design there are different workflows which are of interest for engineers and architects. They accumulate in questions like:

- What are the possible values for a parameter considering all other parameters and their dependencies?
- How can a solution be found which fulfills different criteria and how can these criteria be formulated as a query?
- How can the available solution space be visualized?

Currently there exists no known CAD (computer-aided design) or modeling software which supports set-based design using intervals for the ranges of parameters and constraints for describing the dependencies in geometry. The idea of developing such kind of tools has not yet been realized, but Bettig et al. (2005) state, the benefits would be significant especially in planning processes of the building industry.

The implementation of algorithms to answer the raised questions requires a large effort, but is highly useful for engineering or architectural design tasks. This paper focuses on algorithms needed to compute efficiently the available range of parameters for geometry considering all modeled constraints. As the underlaying problem is the computation of solutions of linear equations – therefore also nonlinear equations – with interval values is NP-hard (Kreinovich and Lakeyev, 1996), this research has the goal of combining existing algorithms which are highly efficient in computing these ranges. The presented solution combines existing algorithms especially from the field of interval computation. It is capable of solving arbitrary systems, can be applied to any constraint graph and is independent of structural properties of the problem in the sense of completeness. This includes systems that are weak or strongly under-constraint as well as well-constraint systems. Linear and non-linear equations are supported. The direction of dependency between parameters has not to be determined as there has no difference to be made between input and output parameters. The presented example is intentionally small, because the complexity of constraint systems usually increases sharply with the number of objects. Relating thereto, it is difficult to describe and to explain plainly the structure of a constraint system. Research in presenting structures of constraint systems is beyond the scope of this paper. This paper has a focus on the computational aspect. It also introduces the idea of set-based design, the geometric constraint satisfaction problem and interval computation which aren’t widespread, especially their combination.

2. Methodology

2.1 Geometric Constraint Satisfaction Problem

Geometric constraint systems - a special case of constraint systems - include geometric objects like points, lines, circles, etc. and geometric constraints like *orthogonal lines*, *parallel lines*, *coincident points*. Geometric constraints describe the relations of geometric objects in a semantical way. This is different to numeric constraints which can be regarded as linear or nonlinear equations. Nevertheless, it's possible to describe geometric constraints as numerical constraints using one or more equations. The description of a constraint system is declarative which means the problem itself is described instead of the so-called imperative approach which prescribes the steps to solve the problem. Because of missing information about the way of solving, this is regarded as the geometric constraint satisfaction problem (GCSP) as a part of the field of constraint satisfaction problems (CSP).

According to Hoffmann and Joan-Arinyo (2005), Joan-Arinyo (2009), Bettig and Hoffmann (2011) and Sitharam et al. (2019) there exist different approaches of solving GCSP: An algebraic approach by solving equation systems; theorem proving by analyzing axioms linked to theorems; a logic-based approach by assumptions and axioms which get solved with a rule-based reasoner rewriting the constraint system to generate procedural solving steps and graph-based by dividing the constraint graph into subproblems and using known solution strategies for the subproblems

Solving algorithms heavily depend on the properties of the constraint system. Various algorithms only solve systems that are serializable (Sitharam et al. 2019, p. 145 and 152). This includes an ordered processing of the constraint graph to create a construction plan or creating a triangular equation system. Complex systems exist which are not serializable thus need a more sophisticated solver. These systems are called "variational" (Sitharam et al. 2019, p. 145) as their parts are mutually dependent. This is usually the case whenever the number of included dependencies is significantly increased.

Another significant property is the existence of solutions. In the case of exactly one solution of the CSP the system is regarded as well-constraint. This is the intended state in a point-based modeling approach. Over-constraint (no possible solution) and especially well-constraint systems can usually be handled by all solving algorithms. In contrast to this, under-constraint systems suffer from the multiple solution problem. This is also referred to as the *root identification problem* regarding CSP represented as an equation system. The problem is the non-existence of a criteria for choosing one solution from the set of solutions. Although Shimizu et al. (1997) state that the user usually sets too less or too many constraints. In this paper it is assumed that under-constraint systems are the intended standard case as the set-based design proclaims. The well-constraint case occurs as the special case when a single final solution is found and the design process finishes.

2.2 Interval Arithmetic

The representation of intervals and their arithmetic was introduced by Moore (1966). There exist various publications, research projects and also a standard (IEEE Std 1788-2015) which is based on the floating-point number standard IEEE Std 754-2008. The main theorem in the calculation with intervals is the inclusion isotonicity of the projection. Every function has to guarantee the inclusion of the solution despite of any rounding errors or discontinuity. Dependent on the function and the variant of arithmetic, the operation is not necessarily bi-total apart from rounding errors. This is usually the case for arithmetics only supporting exactly one

interval as a result for a function. Variants for interval arithmetic can be mainly classified in "traditional" interval arithmetic which results in a maximum of one interval and real numbers as bounds and the extended interval arithmetic which can have infinite bounds and supports results of more than one interval (Kulisch 2013, p. 140). The extended interval arithmetic can handle divisions by intervals including 0 more accurately by introducing more cases for divisions. This is also true for e.g. the square root function.

An important effect in the calculation of intervals is the interval dependency problem. It occurs whenever a variable with an interval value is used more than once in a formula or calculation sequence. For example, using the variable $A = [-2, 2]$ in a calculation of $A^2 = [0, 4]$ leads to a different result than multiplying the variable separately $A \cdot A = [-4, 4]$. The square function considers A to be the same variable, whereas the multiplication does not consider the obvious dependency of the two factors. This leads to an overestimation of the result. The interval dependency problem is the main difference between computation with intervals and with single real numbers, especially in the evaluation of results.

2.3 Interval Constraint Satisfaction Problem

Interval constraint systems are a special case of numeric constraint systems. Every variable is defined for the domain $\mathbb{IR}$ and constraints describe the arithmetic relation between variables. The vector of all values of each and every variable in an interval constraint system is called *box* (Moore et al. 2009, p. 15). A box spans in the n-dimensional space. Different queries arise to an interval constraint system and solving these queries is usually affected by the interval dependency problem. Possible queries are: Finding one arbitrary solution, finding the range of valid values of one or more variables, finding the minimum or maximum of a function also known as rigorous optimization. Searching for the range of all variables is the analogion to searching the ranges of parameters in a geometric constraint system. The quality of a box of values for a constraint system can be categorized by a certain level of consistency.
Lower consistencies, like *arc-consistencies* (Mackworth 1981; Miguel et al. 2001) and *box-consistencies* do not consider multiple occurrences of variables and thus the interval dependency problem. In contrast to these lower consistencies, the more rigorous *hull-consistencies* do consider them. The *kB-consistency* guarantees that the intervals of k-1 other variables are consistent to a variable (Lhomme 1993). This leads to the *(n+1)B-consistency* where *n* is the number of all variables. This is also called the *global hull-consistency* (da Cruz 2003, p. 81) and guarantees that the lower and upper bound of every interval in a box is a solution to the interval constraint satisfaction problem and all other solutions lay in between those bounds. Although, approaches exist which can compute the solution space regarding a certain precision (Fränzle et al. 2007), this paper focus on the global hull-consistency which can be computed with a reduced effort.

3. Existing Approaches

According to Bettig and Hoffmann (2011), computing the valid values of parameters is an open question of GCSPs. A former approach for polygons is presented by Hoffmann and Kim (2001). The presented algorithm is actually only applicable to linear 1-dimensional problems. It computes the range of a single dimensional constraint which is going to be adjusted. The extended approach by van der Meiden (2008) can be used for 2-dimensional cases with distances and angles. His goal is the computation of the valid parameter range of exactly one so-called variant parameter. He uses a constructive graph-based approach by computing special points where the variant parameter has critical values. These are used to determine the minimum

and maximum of the range. Hidalgo Garcia (2013, chapter 4) examined the work of Meiden and proofed that the minimum and maximum can be determined but also disjunctive domains.

An approach for using intervals in modeling and solving GCSPs have been introduced by Nahm et al. (2006). They used a prototype of a CAD software for modeling geometric and non-geometric data in early stages of design under a strong uncertainness. They used an interval propagation algorithm which is strongly limited as it cannot solve equations simultaneously though is not applicable for the general case to create the global hull-consistency.

Wang et al. (2007) also used constraint-based modeling with intervals for geometry objects. They solve the equation system by piece-wise linearization with the Gauss-Seidel method. This can narrow intervals, but suffers from the interval dependency problem and cannot create the global hull-consistency in every case. Their presented example is solvable with the simpler HC4 algorithm developed by Benhamou et al. (1999). Kirchner and Huhnt (2018) used the HC4's sub algorithm HC4Revise to compute the global hull for tree-structured constraint graphs which are proven to be free from the interval dependency problem (Benhamou et al. 1999).

In summary, there is no known approach yet for computing the global hull-consistency efficiently and apply it for the computation of valid ranges of all parameters for a GCSP. These ranges are defined by a lower bound and an upper bound which are each part of at least one solution.

4. Proposed Approach

The presented concept includes the transformation of geometric constraints to numeric equations including all parameters as possible interval values. The graph-based representation of the parameters and constraints is transformed to a set of equations and variables. Each constraint is represented by one or more equations and each parameter represents one variable. An example for a transformed equation is given in figure 1.

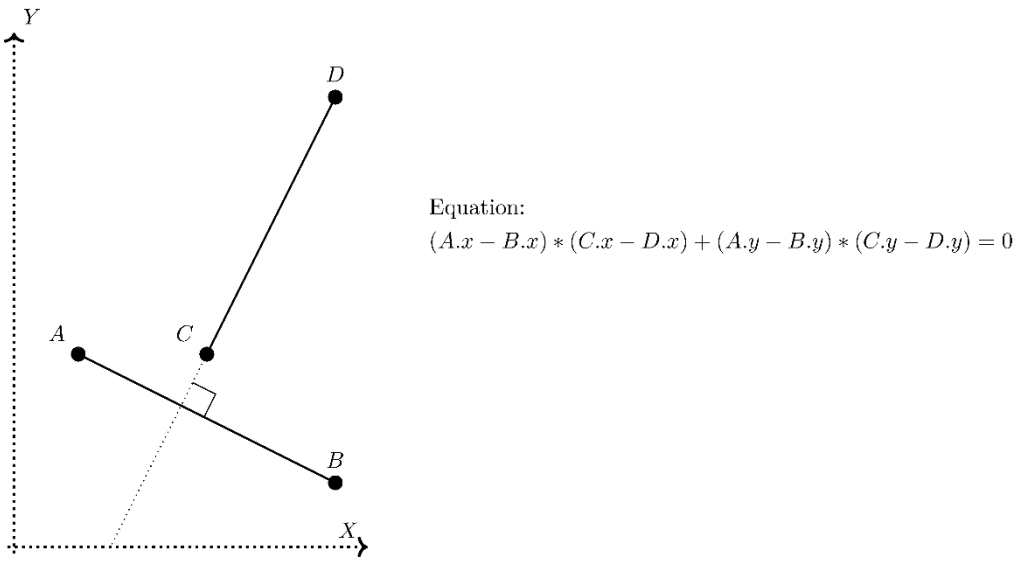

Figure 1: *Orthogonal constraint* for two lines and its equation after transformation.

As common geometric constraints are based on non-linear equations the resulting equation system is non-linear. Additional non-geometric constraints can also be included as equations. It is possible to have more, equal or less variables than equations as this is covered by all chosen algorithms.

The used interval computation library has to support at least the basic mathematical operations and functions like *square* or *square root*. This is the minimal set for the most common geometric constraints. Additionally, set operations like union and intersection are required. For effective solving of interval constraint systems the implementation has to be capable of computing the inverse functions of all operators and functions. An optional requirement which improves the solving process is the differentiation of every equation. This can be realized by automatic differentiation.

The main algorithm to compute the lower and upper bound of each parameter is the *global hull* by da Cruz (2003). This is an efficient algorithm to compute the *(n+1)B-consistency* for an arbitrary case independently of the structure of the constraint graph. The main algorithm uses a set of subalgorithms for reducing the search space. The subalgorithms are heuristics to accelerate the steps to exclude certain parts of the search space which are proven to not include a solution. As there exists a variety of usable heuristics, the proposed approach includes the well-known algorithms *HC4* (Benhamou et al. 1999) and *Interval Newton method*. The Interval Newton method is the interval extension of the Newton-Raphson method considering the first derivative for in- and exclusions.

5. Results

5.1 Implementation Details

The implementation uses a self-developed Java library for interval arithmetic and operations based on the multiple-precision datatype *BigDecimal* of the Java standard. The multiple-precision approach is noticeably slower, but it offers more control over rounding modes and precision in mathematical operations. This is necessary as the control of rounding modes for native IEEE754- floating-point datatypes is not supported by the Java language. The lack of speed is acceptable for research, but is not intended to be used in end-user software. The library also includes the support for extended interval arithmetic which supports more sophisticated case distinction for operations like division or square root. Especially division which is crucial for the *Interval Newton method* leads to disjunctive interval with a reduced overestimation in the intermediate results.

The implemented geometry model is a boundary representation (B-rep) coupled with parameters and constraints. The basic shapes polygon, line and boundary are used to create solids with faces. Only points include a parameter for each coordinate. Extended shapes like e.g. rectangles can be constructed by adding parameters and constraints to basic shapes. Every shape can be assigned to one *functional unit*. Every *functional unit* might represent a digital building element or another element of design or modelling purpose. The objects also can carry a set of parameters which are connected to the parameters of the contained shapes.

A number of geometric constraints is implemented including e.g. parallel lines or orthogonal lines. Additionally, arithmetic constraints for further constraining parameter values are available. They include e.g. the basic mathematic operations and the equality constraint.

Although the parameters, shapes, functional units, constraints and assignment relations are part of a graph representation for analyzation and visualization purposes, a graph is not a requirement for the algorithms. The essential data needed for the computation consists of the equations and variable included in the set of constraints.

5.2 Applied Example

The algorithm was successfully applied to different models like the examples given by Hoffmann and Kim (2001) or Wang et al. (2007). The results correspond with each other.

An example in detail is based on the widely used parametric model of an *extruded rectangular wall*. This is similar to the *IfcWallStandardCase* in the straight case as described by IFC4.2. The wall is described by its *thickness* and *height* and two points forming a line in an XY-plane for placement. These parameters are sufficient for the creation of a geometric representation. The parameter *length* is also added and represents the distance between the two points – P_1 and P_2 – forming the placement line. For a valid solid the points P_3 and P_4 are needed to form the base rectangle with P_1 and P_2 and P_5, P_6, P_7 and P_8 to form the top rectangle. The four side faces are also rectangles formed by pairs of points each from the base and the top. A 3-dimensional sketch is shown in figure 2.

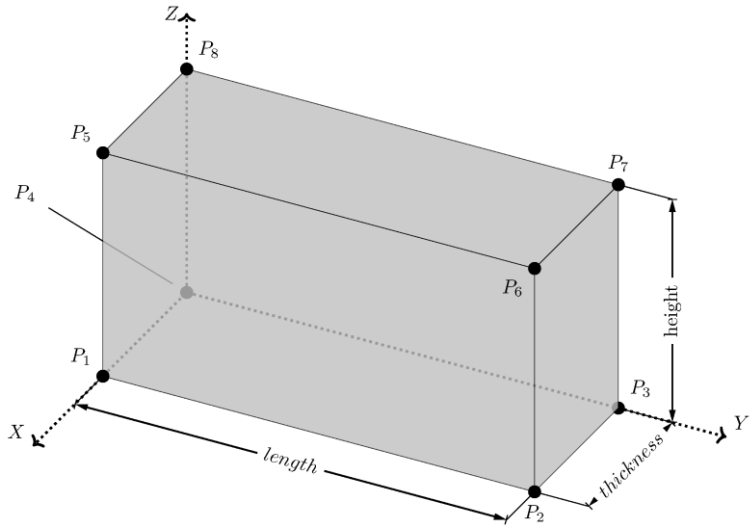

Figure 2: The parameters and vertices of the example *extruded rectangular wall*.

The included constraints for modelling are: orthogonally with oriented distance parameter between line and point in a XY-plane (Orth-O-Dist-XY), oriented distance parameter in z-direction (O-Dist-Z), XY-coordinates of two points equal (Eq-XY), distance parameter of two points (Dist-XY). The constraint graph is shown in figure 3.

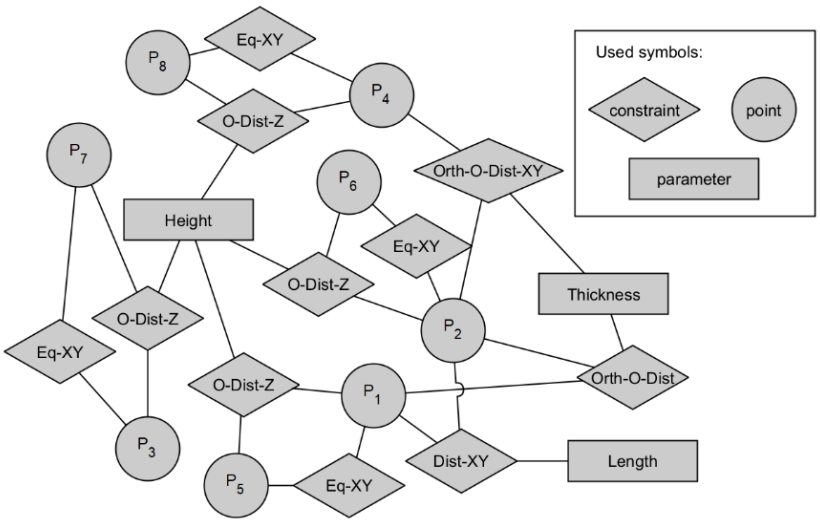

Figure 3: The constraint graph of the example *extruded rectangular wall* including the interrelationships between points, constraints and parameters.

Please note, this is one possible constraint graph. A set of possible variants for constraints in a parametric model of the extruded rectangular wall exists. They share the same parametric behavior.

The example includes 27 variables (3 coordinates for each of the 8 points and 3 parameters for the wall) and 27 equations. A computational run with initial values and the resulting global hull is given in table 1. The computation was limited to the third decimal place.

Table 1: Selected variables and their initial values and their value after computation for the example.

Variable	Initial value	Computed global hull
P_1.x	[3]	[3]
P_1.y	[0, 1]	[0, 1]
P_2.x	[3, 4]	[3, 4]
P_2.y	[5]	[5]
P_3.x	]- ∞, ∞[	[3.39, 3.710]
P_3.y	]- ∞, ∞[	[5, 5.15]
P_4.x	]- ∞, ∞[	[2.39, 2.709[
P_4.y	]- ∞, ∞[	[0, 1.15]
Thickness	[0.3, 0.6]	[0.3, 0.6]
Height	[2.2]	[2.2]
Length	]0, ∞[	[4, 5.099]

The computed global hull corresponds with the expected results. Because of the freedom of P_1 in x- and P_2 in y- direction the length of the wall is computed with the lower bound of 4 and the upper bound of 5.099 as the wall can be placed vertically or slightly diagonally. The possible diagonal placement and $\overrightarrow{P_3P_4}$ always has to be parallel to $\overrightarrow{P_1P_2}$ by a distance of the thickness, results in intervals instead of single values for the coordinates of P_3 and P_4.

6. Conclusion and Outlook

This paper presents a solution on the problem of computing ranges of parameters for arbitrary geometric constraint systems. The computational result is equivalent to the *(n+1)B-consistency*. The solution transforms the geometric constraints to a set of equations and combines existing algorithms for interval computation to find the relevant solutions for the lower and upper bound of ranges. This is approach can handle under-constraint and well-constraint systems for non-linear equations. The results are promising, although a valid comparison to existing approaches is still missing.

This way of modeling using intervals and constraints requires a change in the way on how to create and work with parametric models. It requires the user to approach an intended solution by exploring the design solutions and is similar to set-based design.

The limit of this approach is the computational effort which is unpredictable for complex models as it depends on the used heuristics and the structure of the constraint system. This means a solution to an apparently complex system is found in a short period of time, but there exist simple examples which can take a long time in the worst case. This is usually the case if a large part of the solution space has to be checked, because it can't be excluded early. Another limit is the requirement of partly continuous mathematical functions excluding piecewise-

defined functions. The fixed set of geometry objects for one computation is inherent and limits the applications. The approach does not include models which can have parameters that change the geometric topology e.g. the number and the different types of geometry objects and parameters. This is limited because all constraints have to be transformable to an equation system with a fixed number of variables and equations before the computation can start.

There are different directions of research to pursue. One obvious option is using this approach for parametric studies of different engineering domains. This is promising as it makes it possible to compute valid ranges of input parameters for given intervals of output parameters. Also, bigger models have to be created and get tested. The used algorithms are suitable for concurrent computation, but the performance has not been systematically investigated. Optimizations in code are assumed to increase the computational efficiency noticeably. This also has to be compared to existing interval arithmetic libraries and approaches.

An open question is the applicability of the entire set-based approach. This would require further research on proper visualizations of the solution space, comparison of different solutions and analyzing and querying of constraint systems for certain solutions.

Remarks

This work is a part of a PhD thesis published 2021 only available in German language:
Kirchner, J.V. (2021). Wissensintegrierende Modellierung der Geometrie von Bauwerken mit Intervallen und Constraints für Parameter. TU Berlin. Available at: https://depositonce.tu-berlin.de/handle/11303/12690

References

Benhamou, F., Goualard, F., Granvilliers, L. and Puget, J.-F. (1999). Revising Hull and Box Consistency. In: Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999. MIT press, pp. 230–244.

Bettig, B., Bapat, V. and Bharadwaj, B. (2005). Limitations of parametric operators for supporting systematic design. In: Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - DETC2005, 5. pp. 131–142.

Bettig, B. and Hoffmann, C.M. (2011). Geometric Constraint Solving in Parametric Computer-Aided Design. In. J. Comput. Inf. Sci. Eng. 11, 021001 (9 pages).

da Cruz, J.C.F.R. (2003). Constraint reasoning for differential models. PhD Thesis. Universidade Nova de Lisboa, Lisboa.

Fränzle, M., Herde, C., Teige, T., Ratschan, S. and Schubert, T. (2007). Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure. In: Journal on Satisfiability, Boolean Modeling and Computation 1(3–4). pp. 209-236.

Gil, N., Beckman, S. and Tommelein, I.D. (2008). Upstream Problem Solving Under Uncertainty and Ambiguity: Evidence from Airport Expansion Projects. In: IEEE Transactions on Engineering Management 55. pp. 508-522.

Hidalgo Garcia, M.R. (2013). Geometric constraint solving in a dynamic geometry framework (PhD Dissertation). Universitat Politècnica de Catalunya, Barcelona.

Hoffmann, C.M. and Kim, K.-J. (2001). Towards valid parametric CAD models. In: Computer-Aided Design, 33(1), pp. 81–90.

Hoffmann, C.M. and Joan-Arinyo, R. (2005). A Brief on Constraint Solving. In: Computer-Aided Design and Applications 2. pp. 655–663.

Institute of Electrical and Electronics Engineers [IEEE] (2008). IEEE 754-2008 - Standard for Floating-Point Arithmetic.

Institute of Electrical and Electronics Engineers [IEEE] (2015). IEEE 1788-2015 - Standard for Interval Arithmetic.

Joan-Arinyo, R. (2009). Basics on Geometric Constraint Solving. In: Proceedings of 13th Encuentros de Geometría Computacional (EGC09). Presented at the Encuentros de Geometría Computacional (EGC09, Zaragoza (Spain).

Kirchner, J. and Huhnt, W. (2018). Benefits and Limits of Interval Constraints in Acyclic Constraint Graphs for Geometric Modeling. In: Proceeding of the 17th International Conference on Computing in Civil and Building Engineering. Computing in Civil and Building Engineering (ICCCBE) 2018. Tampere, Finland. pp. 716–723.

Kreinovich, V. and Lakeyev, A.V. (1996). Linear interval equations: Computing enclosures with bounded relative or absolute overestimation is NP-hard. In: Reliable Computing, 2(4). pp. 341–350.

Kulisch, U. (2013). Computer Arithmetic and Validity, Theory, Implementation, and Applications. De Gruyter, Berlin, Boston.

Lhomme, O. (1993). Consistency Techniques for Numeric CSPs. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'93. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 232–238.

Liker, J.K., Sobek, D.K., Ward A.C. and Cristiano J.J. (1996). Involving Suppliers in Product Development in the United States and Japan: Evidence for Set-Based Concurrent Engineering. In: IEEE Transactions on Engineering Management 43 (2), pp. 165–178.

Mackworth, A.K. (1981). Consistency in Networks of Relations. In: Webber, B.L., Nilsson, N.J. (Eds.), Readings in Artificial Intelligence. Morgan Kaufmann, pp. 69–78. https://doi.org/10.1016/B978-0-934613-03-3.50009-X

Miguel, I. and Shen, Q. (2001). Solution Techniques for Constraint Satisfaction Problems: Foundations. In: Artificial Intelligence Review. 15(4), pp. 243–267.

Moore, R.E. (1966). Interval analysis. Prentice-Hall, Englewood Cliffs, NJ.

Moore, R.E., Kearfott, R.B. and Cloud, M.J. (2009). Introduction to Interval Analysis. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Nahm, Y.-E. and Ishikawa, H. (2006). A new 3D-CAD system for set-based parametric design'. In: The International Journal of Advanced Manufacturing Technology, 29(1–2). pp. 137–150.

Shah, J.J. (2001). Designing with Parametric CAD: Classification and comparison of construction techniques. In Geometric Modelling. New York: Springer Science+Business Media, pp. 53–68.

Shimizu, S. and Numao, M. (1997). Constraint-based design for 3D shapes. In Artificial Intelligence, Artificial Intelligence Research in Japan 91. pp. 51–69.

Sitharam, M. and St. John, A., Sidman, J. (2019). Handbook of geometric constraint systems principles. CRC Press, Taylor & Francis Group, Boca Raton.

Valdes, F., Gentry, R., Eastman, C. and Forrest, S. (2016). Applying Systems Modeling Approaches to Building Construction. In: Proceeding of the 33th International Symposium on Automation and Robotics in Construction. Presented at the 33th International Symposium on Automation and Robotics in Construction, Auburn, AL, USA.

van der Meiden, H.A. (2008). Semantics of Families of Objects. PhD Thesis. Technische Universiteit Delft.

Yannou, B., Yvars, P.-A., Hoyle, C. and Chen, W. (2013). Set-based design by simulation of usage scenario coverage. In: Journal of Engineering Design 24, pp. 575–603.

Wang, Y. and Nnaji, B.O. (2007). Solving Interval Constraints by Linearization in Computer-Aided Design. In: Reliable Computing, 13(2). pp. 211–244.