Modelling with Constraints: A Bibliographical Survey

Ghassan Kwaiter, Véronique Gaildrat, René Caubet

Department of Computer Science I.R.I.T
Paul Sabatier University
118, route de Narbonne, 31062 Toulouse Cedex, France

Phone. (33) 5 61 55 83 29
Fax. (33) 5 61 55 62 58
E-mail: kwaiter@irit.fr

Abstract

Nowadays, the constraint solvers play a fundamental role in many graphical applications, including geometrical and mechanical CAD systems, layouts, user interface, and animation. These applications use constraints to conveniently describe relations between objects. Constraints are automatically maintained and satisfied by a constraints solver. So, numerical and tedious tasks are left to the solver and the designer can concentrate on design process.

In this paper, we present concepts of constraints and constraints solvers. We classify the principal resolution methods, giving their drawbacks and shortcomings. We also examine different domains of applications using constraint solvers.

1. Introduction.

Many aspects of interactive graphical applications can be conveniently described using called constraints. Constraints describe relations that must be maintained by delegating to the constraints solver the task to satisfy them automatically. Their usefulness stems from the fact that constraints provide a useful declarative mechanism to help the designer in graphical applications and emphasize the relations itself rather than technique used to maintain them.

For instance, in intelligent CAD applications, constraint-based systems are increasingly being used to support design of product. Several design applications based on constraints allow the geometry of a product to be specified and modified in a more natural and efficient way. The declarative semantics of constraint languages allows one to specify graphical objects and their inter-relationships while avoiding extraneous concerns about the realization of the visualization algorithms. Indeed, applying constraints in the design allows a better control and facilitates an incremental re-design of a generated presentation.

The paper is organised as follows: Section 2 introduces concepts of constraints and constraints solvers, and describes how constraints can be used in many graphical application areas. Section 3 characterises the fundamental constraints solver specifications to should be able to satisfy several designers' requirements. Section 4 presents different classes of constraints solver methods. In Section 5, a wide range of related constraint-based systems are described. Finally, Section 6 presents our conclusion.

2. Constraints and Constraints Solvers.

In some limited graphical applications, it is possible for the designer to include a code that maintains *implicitly* several relationships among objects. For example, a *Callback technique* can be used to satisfy constraints when a particular event occurs. On the other hand, as the system grows up and more complicated, it becomes much more difficult to keep track of all these relationships. Changing one relation in such a system may involve modifying the whole code.

Alternatively, many relationships among objects in design applications can be expressed *explicitly* as constraints. Normally, a *constraint* is a relation that the designer wants to be satisfied. In other words, constraints represent the desired relationships among objects. In mathematical term, a constraint defines a relation among a set of variables involved into the relation. Given a set of constraints, an underlying mechanism is invoked to find the attribute values to satisfy these constraints. Such a mechanism is called *constraints solver*.

A constraints solver is a generic term used to describe a framework that allows the specification of constraints and ensures they are satisfied. Many

advantages to use constraints instead of other simpler consistency mechanisms can be founded:

- Constraints represent the knowledge domain and the functionality of objects. They can also define and control the space of research, and evaluate the possible solutions.
- By assigning this responsibility to the constraints solver, the code of programs can be much simpler, and the relationships are easier to manipulate and to maintain than a collection of code fragments. Moreover, a generic constraints solver should be application independent to be used in different graphical applications.
- The declarative semantic of constraints allows to specify relations among objects and a constraints solver allow the designer to concentrate on the relationships that need to be maintained, rather than on the technique used to maintain them. So, applying constraints in a design allows a better control and facilitates an incremental design of a generated presentation.
- A constraints solver has an explanatory nature: the designer could be able to examine the system's state to be informed of satisfied constraints, and why some constraints become unsatisfied.

These advantages explain why constraint-based systems have shown usefulness for interactive geometric design. For instance, geometric shape or object's functionality are specified by constraints such as distances, angles, incidence, adjacency, tangency, etc. and a constraints solver can be used to derive the shape from such a specification or to describe this functionality.

3. Constraints-Based Graphical Applications Requirements.

The ability to represent and maintain relationships among objects can be an extremely useful tool in a graphical application. Since the earliest interactive graphical application [77], usefulness of such constraint techniques has been demonstrated in application including drawing, 3D modelling, user interface construction, animation, and design.

With constrained-based techniques employed in such graphical applications, system designers face a variety of new additional challenges. In addition to graphical objects, constrained models also contain constraints that must be stored, displayed, edited, saved, etc. Thus, constraints introduce challenges in implementation, in performance, and in usability. Thus, a truly constraints solver should satisfy several requirements:

- Generality: In graphical applications restricting the type of constraints is unacceptable. In the most basic graphical application linear and non-linear constraints are required. Moreover, inequality constraints are useful to specify various aspects of user interface and graphic applications such as layout and topologic constraints. Thus, constraints solver must support several kinds of constraints.
- Dynamically: One important technique in interactive applications where the relationships between objects can be changed dynamically. In these applications, the designer may want add more constraints between objects, or may want remove constraints to release constrained objects. Moreover, execution time must take advantage of every computational opportunity to improve performance. For all these requirements, a constr-aints solver must dynamically and quickly re-satisfy the system.
- Efficiency: In interactive graphical application, it is not acceptable for the constraints solver to halt, signalling an error or crashing when confronted with a set of constraints that cannot be satisfied simultaneously. Such a failure occurs either when a solution cannot be found because of conflicting constraints, or with under-constrained problem. Thus, it is important for the constraints solver to efficiently manage these critical situations.
- Complexity: Constraints describe object behaviour. Each time the user adds a constraint among objects, these objects will have a new own behaviour. Thus, behaviours are specified in terms of constraints. Complexity of constrained behaviour in a graphical model becomes a problem when the behaviour is not what expected. This arises either when constraints give a configuration that the designer does not wants, or constraints prevent the designer from achieving a desired configuration.
- Performance: Performance is important for graphical applications, particularly in animation to achieve the appearance of continuous motion. In much application, objects can potentially move simultaneously, thus the complexity becomes higher for the constraints solver.

For instance, in direct manipulation applications, designers may need to smoothly explore the designing domain by interactively making local modifications. So, time consuming must take advantage of every computational opportunity to improve performance. The constraints solver must dynamically maintains constraints as the user modifies the design and the solver must quickly give a unique solution even in critical situations such as over-constraints or cyclic constrains problems.

4. The Range of Constraints Solvers.

In selecting the kinds of constraints to be supported in graphical systems, there are various tradeoffs between simplicity and efficiency, according to the problem's complexity. Some of these are numeric constraints, or other domains; one-way or multi-way constraints; functional constraints only or more general relations; required constraints only, or constraint hierarchies; acyclic constraint graphs only, or cycles allowed. Solvers can also be compared according to their behaviours: when all constraints cannot be maintained, several solutions and conditions are suggested, especially the used methods to handle dynamic and incremental constraints.

The various approaches strongly depend on the structure and the complexity of the specific application domain. In this section we survey the major approaches and we characterise their advantages and shortcomings in graphics systems.

4.1 Numeric solvers:

Numeric solvers intent to solve constraints by translating them into a system of algebraic equations. A number of iterative numeric techniques, such as classical relaxation, Newton-Raphson, can be used to satisfy sets of linear and non-linear constraints. For instance, Relaxation has been used in Sketchpad [77], the first 3D interactive application. Newton-Raphson iteration is used in Converge [73] and Juno [65]. Frequently, iterative numerical techniques are combined with local propagation solvers to solve cycles or simultaneous constraints. Although with these techniques, a designer can easily add or remove a constraint, iterative numeric techniques have several drawbacks that make them difficult to apply in interactive applications: They can be very slow to converge as the number of constrained variables grows larger, and they may not converge on a solution that satisfies the constraints. Further, the solutions produced can be very sensitive, and irregular. Moreover, these methods do not offer any efficient solution if the system is over-constrained.

4.2 Local Propagation Solvers:

Local propagation technique is the most commonly used constraint solving technique in graphical applications such as Sketchpad [77] and ThingLab [8]. This technique represents constraints by a set of *method* procedures. These methods access some of constraint's variables and calculate values for remaining variables in order to satisfy the constraint. When the values of one or more variables change, the solver dynamically attempts to direct the graph to solve every constraint by executing its selected method. There are two main kinds of local propagation solvers: one-way and multi-way. One-way solvers [41,44, 64] allow each constraint to have a single method that satisfies the constraint in one direction. Multi-way solvers [30,72,80] allow a constraint to have multiple methods, used by the solver, to satisfy the constraint in different directions.

Local propagation solvers can be optimised by making them incremental, so only the elements affected by changes are computed. Efficient algorithms that compute minimal numbers of dependencies include the DeltaBlue [30] solver and its successors such as SkyBlue [72]. These solvers have another interesting property; they are *hierarchical* [9]: They permit declaring certain constraints to be more important than others. So, a strength value can be associated to a constraint to obtain a constraint hierarchy. Thus, the more important constraints are first solved. In case of conflict, less important constraints are used only when more important constraints leave unsatisfied. Local propagation constraints solvers have several advantages, particularly for animation, user interface, and interactive applications:

- Local propagation solvers are very general. A method can perform an arbitrary computation and is not limited to simple algebraic expressions. Constraints solvers based on this technique are able to handle a wide range of graphical applications.
- In many situations, a local propagation constraints solver has the possibility to save a sequence of selected methods in a plane to execute it several times. This propriety is fundamental in animation and direct manipulation applications. For example, when one or several variable are modified in a continuous way, the solver based on local propagation technique reacts very quickly.
- Using multi-way constraints is essential for some interactive applications, particularly for virtual reality application, because of strong connecting between interaction and application objects.

Recently, solvers based on local propagation technique present the best compromise between the efficiency and the extensibility. However, impossibility to handle cycles and inequation constraints are major drawbacks of local propagation algorithms.

4.3 Deductive methods:

Deductive methods take advantage of construction rules and properties of Euclidean geometry in order to establish, by reasoning, a construction model using constraints. Constraints and further deductions are placed in a basis of rules generally expressed in PROLOG [11], or by an expert system [75]. Another approach uses a set of geometric construction rules and a terminated set of rewriting rules [12]. Each construction rule is transformed into a rewrite rule. In

other words, the whole constraint is transformed into primitive terms using rules. Utilisation of a rewrite system allows demonstrating some interesting properties like termination: every rule can be rewrite by successive derivation of primitive terms. General constraints-based languages using rewriting solvers have also been used in graphical applications context, for example Siri [42], Bertrand [54], and Equate [84] systems.

Although a Logic Programming style is a good approach for prototyping and experimentation, the extensive computations when searching and matching rewrite rules constitute a drawback. Furthermore, a rewrite system limits possibilities of extension by dynamically adding or removing constraint. So, this method is not adapted for interactive application.

4.4 Constructive Solvers:

This class of constraints solvers is based on the fact that most configurations in an engineer drawing are solvable by ruler, compass and protractor, or by using another less classical collection of construction steps. In these methods, constraints are constructively satisfied using geometric elements in two orders: descending order [67] and ascending order [75]. For both techniques, the graph of constraints is analysed to found a sequence of construction steps. This sequence is then carried out to derive the solution. For these solvers, primitive constraints are based on distances and angles. Primitive objects are points, lines and circles. Thus, it is very difficult to describe construction phases of 3D objects using this method.

4.5 Symbolic Solvers:

Constraints are transformed into a system of algebraic equations. The system is solved with symbolic algebraic methods, such as Gröbner's base [10], or Wu-Ritt method [19,85]. Both methods can solve general non-linear systems for algebraic equations. These methods have also been used in mechanical geometry theorem proving. In [47], Kondo considers addition and deletion of constraints by using the Buchberger's algorithm [13] to derive a polynomial equation that gives relationships between deleted and added constraints.

4.6 Finite-Domain Solvers:

A particular kind of constraints solving problem extensively studied within the *Artificial Intelligence* community is known as *Constraint Satisfaction Problem* (or CSP) [59]. This approach offers a general representation allowing formulating different types of combinatorial problems, like geometric placement. The innovative concept of finite-domain constraint solving in graphic applications lies in the active use of constraint propagation technique. This technique is used to find a locally consistent solution by pruning the search space and removing combinations of values that cannot appear together in a solution. Then, the solver tries to find one or multiple solutions using different techniques, including intelligent backtracking, looking-ahead and forward checking technique [69].

Particularly, in numeric CSP model, functional, inequality, linear, and non-linear constraints are supported, and variables are represented by an interval of continuous numerical values. In this area, we can distinguish several works on interval constraints. Classical propagation algorithms for continuous domains such as Waltz propagation [81] and Davis algorithm [23] provide relatively poor results: they ensure neither completeness nor convergence. Interval propagation technique is used by Hyvönen [45] to compute local consistency in term of *solution functions*.

However, these techniques are not always able to determine the variable domain: slow to converge and limited for disjunctive constraints. Furthermore, domain-splitting technique introduces choice point problem and can quickly leads to combinatorial explosion. Lhomme [55] proposes interval propagation formalism based on bound propagation. Another active area of research is the incorporation of interval constraints into logic programming. Examples of such systems include CLP (BNR) [5], Newton [6] and Cleary [20]. Important shortcoming in these algorithms is the postulate that constraints are required and static, thus, dynamic and hierarchical concepts are not supported. In order to surmount these drawbacks, Kwaiter et al. [49, 50] have presented a generic constraints solver called ORANOS. This solver uses an interval propagation technique to maintain equality and inequality constraints. ORANOS is a dynamic constraints solver, that dynamically re-satisfies the constraints network as new constraints are added, existing constraints are removed, and constraint strength is changed. It supports constraint hierarchy in order to deal with over-constraint problem.

5. Constraints-Based Graphics Systems.

Many systems use constraints solvers to create and maintain geometric relationships between graphic objects. The following subsections describe some of them, and examine techniques developed to specify and represent constraints. Since there is a long history of using constraints for graphical applications, we concentrate rather on system that include dynamic and hierarchies constraints solver.

5.1 Drawing Systems:

In a constraint-based drawing system, the user specifies relationships among drawing parts as persistent constraints maintained by the system during later edition. In case of computer-aided design systems,

modifying these relationships may be a major part of the design.

Major interactive drawing applications support geometric constraints among graphic objects. Sketchpad [77] pioneered direct manipulation permitting users to directly manipulate graphical objects by dragging them with the light pen and also was used in order to feign some mechanical links. Sketchpad introduces constraint methods, permitting users to specify relationships between objects; for example two lines must always be parallel. Users can drag objects with a light pen in order to build mechanical links. The resolution takes place in two phases. First, propagation of the degrees of liberty, and second, a relaxation phase based on numeric methods used if the first phase fails because of cycles.

Sussman and Steel [76] have implemented several constraint languages distinguished by their ability to maintain dependency information. Then, this information can be used both to produce explanations about derived values and to direct backtracking. However, the interaction mode is strictly textual. Vandyke's IDEAL system [78] is a language to compose graphics that uses constraints to describe graphic objects and their relative positions. Gosling [36] has produced a constraint-based graphical layout system called Margritte. Margritte's constraint system operates only on simple numeric scalars. However, the user may define new structures and complex constraints to operate on these structures. Both IDEAL and Margritte use algebraic techniques to solve simultaneous linear equations.

The graphic system Juno [65] allows constraints to be graphically specified by selecting icons and points. These constraints relate model elements to a textual program in charge of drawing the model. However, the system domain is limited to simple primitive objects. Contrary, Juno2 [39] provides a powerful declarative language in order to enlarge its extensible class of non-linear constraints. CoolDraw [32] is a constraint-based drawing system that uses a local propagation technique [72] to maintain geometric relationships between objects in 2D. In CoolDraw, the user can create, move, reshape, and delete objects. In addition the user can dynamically add and removes constraints on objects.

Constraint techniques have also been applied in 3D systems. The Variational Geometry system of Light and Gossard [57] has restored interest in use of constraint techniques for designing 3D objects. Bruderlin [11] and Rossignac [71] have presented constraint based solid modellers. Bramble [34] supports drawing in three dimensions. In this system, lights and cameras are considered as objects to be constrained and manipulated. Constraint methods have been applied to surface modelling, allowing users to manipulate surfaces without knowing the underlying representation. Fowler [29] and Welch, Gleicher and Witkin [82] present a simple constraint method to control point on B-Spline surfaces. Celniker [14] describes methods to interactively optimise a shape. In [15], this method has been extended to a larger class of constraints. Welch and Witkin [83] extend this work to a wider variety of constraints to permit to a user to stitch together pieces of surfaces.

5.2 Constraint-Based Direct Manipulation Systems:

Direct manipulation techniques have been successfully used for geometric modelling tasks. With these techniques designers, for example, can control the geometry of objects by interactively garbing and pulling them, with continuous update. In this area, we can distinguish several works that support interactive user manipulations:

Bier [7] has presented a technique developed to quickly and interactively design precise 2D and 3D shapes. A synthesis of the best properties of grid-based systems, constraint networks, and drafting has resulted in a new technique called *snap-dragging*. To help precise construction, a set of lines, circles, planes, and spheres, called *alignment objects* are constructed by the system at a set of sloped, angles, and distances are specified by the user. The user can snap the cursor onto these alignment objects, onto their points or lines of intersections, or onto scene objects using an adjustable *gravity mapping*.

The Bramble graphical toolkit [34] supports user interactions through differential manipulation technique [33], where constraints specify the object attributes derivatives over time, rather than their exact values. For example, if the user wanted to drag an object using the mouse, this operation would be implemented with a constraint that force the object to always move towards the current mouse position. This result in a user interface where objects move smoothly as the user interacts with them.

Fa [27] has proposed an interactive constraint-based solid modelling system using an intuitive 3D graphical interface. The system supports constraint-based 3D manipulation techniques used within a virtual environment to design and assemble solid models only with 3D manipulations. The system employs an automatic constraint recognition technique, able to recognise assembly relationships and geometric constraints (such as against, coincidence, tangency and concentricity). Constraints are expressed between solid models from user's 3D manipulations. These constraints are stored in a directed graph called a Relationship Graph (RG). Then, an allowable motion technique computes the remaining degrees of freedom of a solid model from its constraint information in the RG. This allowable motion is used to automatically constrain the subsequent 3D manipulations of the solid model, without invalidating its associated constraints. The system uses graph-based algorithms and incremental constraint satisfaction to allow interactive simulation of inverse kinematics and closed-loop mechanism problems.

TWEAK [43] is an interactive constraint-based manipulator tool set for editing CAD models. It provides manipulator tools to place vertices, planes and rigid objects interactively picked by the user on graphic elements. These manipulators are connected to a *Conceptual Engine 3D* constraints solver, based on propagation of degree of freedom. The solver ensures that changes are consistent with relationships defined between the geometric elements of the model. Moreover, users are not obliged to completely specify shapes by constraints but can freely mix constraint definitions with geometric constructions in a more intuitive manner.

Kwaiter et al. [48] have proposed a geometric constraint system named LinkEdit that provides an interactive tool to construct objects from rigid primitives, and to constrain them using several constraint types. A constraint is presented by a *multi-method* to easily define the reciprocal relations between objects. LinkEdit uses local propagation technique [72] and propagation of degrees of freedom technique [80] to efficiently maintain constraints organised in a constraint hierarchy. As the proposed solver is independent of the space dimension, it is used for both 2D and 3D geometric constraint systems and algebraic ones. In LinkEdit, geometrical constraints, dynamic constraints, and interaction constraint are provided. For instance, when a user selects an object, one interaction constraint is added in the constraint graph and the solver re-satisfies the graph. As the mouse is moved when grabbing constrained object, only a set of local modified methods will be re-executed. However, where the user releases the mouse, the added interaction constraint is removed from the graph. Furthermore, internal and external constraints are introduced as a new paradigm to define the complete behaviour for a rigid object.

5.3 Constraint-Based User Interface Systems:

A number of user interface toolkits allow the programmer to create constraints about graphic objects' position, and to establish connections between different user interface elements.

ThingLab [8] is a constraint-based laboratory used to construct simulations of such things as electrical circuits, mechanical linkages, and demonstrations of geometric theorems. ThingLab use two kinds of local propagation, as well as relaxation, to solve constraints. ThingLab`II` [60] supports constraint hierarchy, and includes a compiler to optimise structured and constrained objects. Optimisation is obtained by discarding unnecessary structures and compiling constraints into native code [31].

Garnet [64] provides many facilities to help the programmer construct highly interactive user interfaces. It includes a prototype-based object system and supports a retained graphics model that eliminates the need for the programmer to explicitly redisplay graphic objects.

Rendezvous system [41] supports distributed applications development, operated by multiple users at once. It is based on the *Abstraction Link View* architecture, where multiple people interact with different graphic displays, co-ordinated via constraints stored in a single database. The user interface construction tools TRIP3 [63] map abstract objects and relations onto sets of graphical items and geometric relations for visualisation.

In virtual reality system Gobbetti and Balaguer has proposed a VB2 system [35] based on SkyBlue constraints solver. The system uses constraints to maintain relationships between object's properties, to maintain connections between the 3D input devices and objects in the virtual world, and to maintain the connection between a virtual tool and an object. For example, at different times the same 3D input device can be connected to different virtual world objects, but removing the current constraint and adding a different one, taking advantages of SkyBlue's ability to rapidly add and removes constraints.

Other user interface toolkits that use constraints include GROW [3], MEL [40], Cactus [62], and GITS [66].

5.4 Constraint-Based Animation Systems:

Several constraint-based systems have incorporated the notion of time. The Animus system [25] supports constraints relating numeric variable values and the derivatives of these values over time. During an animation execution, the derivatives are used to repeatedly compute the next variable value, performing an approximate integration.

TBAG system is a toolkit to create interactive and animated 3D graphics [26]. TBAG supports static objects that represent a single object state, as well as *constrainables* that represent a continuous flow of object states over time. TBAG uses a local propagation technique [72] to maintain constraints. Multi-way constraints can be declared among constrainables. More complex constraints support time-based relationships. A constrainable vector that represents the derivative over time of a constrainable position gives an example of relationship. TBAG includes cycles solver that accepts cycles of constraints, and call an *ODE* solver to satisfy them

Virtual Studio [2] is a 3D animation environment where all interactions are done in 3D and where multi-track animations are defined by recording user's manipulations on 3D models. At the end of the recording session, animation tracks are automatically updated to integrate the new piece of animation. Animation components can be easily synchronised using constrained manipulation during playback. The system uses a multi-way and multi-output dataflow hierarchies' constraints solver [72]. Indirect constraints are exploited to define most of the modelling and animation

components' behaviour in a declarative way. The solver also provides necessary fixed connections among components to compose animated and interactive behaviours.

5.5 Geometric applications as CSP:

Many problems in researches including automated geometric layout, computer vision, simulation, planing and design (which previously appeared to be involved in a particular application domain, and therefore solved in an ad-hoc manner), can now be seen as instances of Constraint Satisfaction Problem technique. For instance, Faltings [28] motivated a constraints-based approach to CAD, Haroud [38] deals with global consistency for continuous domains.

Aggoun et al. [1] has proposed a cumulative constraint system to improve the efficiency of CLP languages for solving difficult scheduling and positioning problem. Baykan and Fox [3] employ the constraint-directed search as a principal solving technique for intelligent CAD, and use some heuristics and criteria to reduce research domain. Criteria who participate to the decision choice are the number of constraints carrying on a variable, the dependence of the constraints and the toughness of the constraints.

Several works have addressed the problem of solving geometric layout as CSP. For example, Charman and Trouss [17] deal with rectangle, whereas Lorenzo and Pérez [58] treats more general polygons, and Du Verdier's technique deals with non-overlapping constraints.

LayLab Framework [37] is a multimedia layout toolkit designed to automate the generated material layout as well as constraint-oriented support of graphical editing and design. It incorporates a collection of two constraints solver including a local propagation solver SIVAS+ and a finite domain solver FIDOS. The first one extends the constraint hierarchy solver [30] by indirect reference constraints in order to allow dynamic input and user interactions. The second, FIDOS interprets positioning heuristics as domain specific constraint abstractions and uses intelligent backtracking to enable an early pruning of the search space.

5.6 Declarative Modeller Systems:

The objective of declarative modelling is to provide some powerful tools to stimulate the designer's creativity during the conception process. With this approach, in description phase, the designer has the possibility to describe a scene without giving numerical data, but by expressing the mental image of the scene in almost natural language. Then, in generation phase, the modeller has to explore all the possible scenes to select the most appropriate.

In declarative modelling, constraints have a very significant role in the design process. The designer uses them to easily describe complex scenes. The declarative semantic of constraints allows to specify relations between objects. Moreover, constraints lead to a better control of the solutions produced by a declarative modeller, to facilitate an incremental design. Thus, the ergonomy of the modeller is greatly improved by the constraints solver efficiency.

In current literature, considerable works have adapted declarative modelling approach according to several points of view. A scene can be described in one step [18, 21, 24, 70], or in an incremental manner [56]. In order to refine the produced solutions, Martine and Martin [61] propose a technique to sort the *representative solutions*, so the user must not visualise all the solutions. Colin and al. [22] provide a set of tools to help the user to understand the produced solutions. Other works address the CSP technique, either to generate one or multiple solutions [56], or to reduce execution time [16,68].

DEM^{2}ONS declarative modeller (Declarative Multimodal MOdeliNg System) [51] represents an interesting declarative modeller for 3D scenes. DEM^{2}ONS allows the designer to easily describe the scene using geometric and topologic constraints in a rather natural language. It also lets him explore the scene from multiple points of view. DEM^{2}ONS accepts to dynamically add and remove constrained objects and launches the incremental constraints solver, ORANOS, to re-satisfy the system when a designer's intervention occurs. Unlike other classical declarative modellers, DEM^{2}ONS [52, 53] finds a non collided position for constrained objects, dealing with semantic and the pragmatic constraints.

6. Conclusion.

Modelling with constraints is a paradigm, which contains a high potential for playing that important role. The primary goal of this paper is to demonstrate the power of constraints as a method to specify relationships among objects in graphical applications, and how the modern CAD/CAD systems support constraint solvers as design automation tools. The second goal is to explain how these solvers are developed to improve the productivity of designers in creating new product design, reuse previous designs, and exploring variations.

Despite its promise and the many research efforts made in the past thirty years, constraint solving has remained a very difficult problem. Even though there are several different approaches developed, there is still no one general, efficient, and robust enough for commercial use.

References

[1] Aggoun A. and Belddiceanu N. Extending CHIP in order to solve complex scheduling and placement

problems. In *Mathematical and Computational Modelling*. Vol. 17(7): pp. 57-73, 1993.

[2] Balaguer JF. and Gobbetti E. Supporting Interactive Animation Using Multi-way Constraints. In *Proceedings of Eurographics Workshop on Programming Paradigms in Computer Graphics*, September, Maastricht, 1995.

[3] Baykan Can A., Fox M. and WRIGHT S. A Constraint based Spatial Layout System. In *Artificial Intelligence in Engineering Design*, Vol. 1: pp. 395-432, Academic Press, 1992.

[4] Barth P. An object Oriented Approach to Graphical Interfaces. In *ACM Transactions on Graphics*, Vol. 5(2): pp. 142-172, 1986.

[5] Benhamou F. and Older W. Applying interval arithmetic to real integer and Boolean constraints. In *Journal of logic Programming*, 1996.

[6] Benhamou F., McAllister D. and VanHentenryck P. CLP(Intervals) revisited. In *International Symposium on Logic Programming*, pp. 124-138, Ithaca, 1994.

[7] Bier E. and Stone M. Snap-Dragging. In *Proceedings of Computer Graphics, SIGGRAPH'86*. Vol. 20(4): pp. 233-240, 1986.

[8] Borning Alan. The Programming Language Aspects of ThingLab, A Constraint-Oriented Simulation Laboratory. In *ACM Transactions on Programming Languages and Systems*. Vol. 3(4): pp. 353-387, October 1981.

[9] Borning A., Freeman-Benson B. and Wilson M. Constraint hierarchies. In *Lisp and Functional Programming*, Vol. 5: pp. 223-270, 1992.

[10] Brown D. H. Associates. Conceptual Design: Tradeoffs in Performance and Flexibility. Notes on the design of Pro/ENGINEER, 1991.

[11] Bruderlin B. Constructing Three-dimensional Geometric Objects Defined by Constraints. In *Proceedings of the ACM SIGGRAPH Workshop on Interactive 3D Graphics*, pp. 111-129, 1986.

[12] Bruderlin B. Using geometric rewrites rules for solving geometric problems symbolically. In *Theoretical Computer Science*, pp. 291-303, 1993.

[13] Buchberger B. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. In N.K. Bose, editor, Multidimensional Systems Theory, pp. 184-232. D. Reidel Publishing Co., 1985.

[14] Celniker G. and Gossard D. Deformable curve and surface finite-elements for free-form shape design. In *Proceedings of the SIGGRAPH'91*, pp. 257-266, 1991.

[15] Celniker G. and Welch W. Linear constraints for deformable b-spline surfaces. In *proceedings of the Symposium in Interactive 3D Graphics*, pp. 165-170, March 1992.

[16] Champciaux L. Declarative Modelling: speeding up the generation. In *proceedings of the CISST'97*, Las Vegas, Nevada, June 30-3July 1997.

[17] Charman P. and Trousse B. EAAS: environnement d'aide à l'aménagement spatial. In EC2, Editor, Avignon'93, Vol. 1, mai 1993.

[18] Chauvat D. Le projet VoluFormes: un exemple de modélisation déclarative avec contrôle spatial. Thèse de Doctorat, Nantes, 1994.

[19] Chou C.S. Mechanical Theorem Proving. D. Reidel Publishing, Dordrecht, 1987.

[20] Cleary J.G. Logical Arithmetic. In *Future Generation Computing Systems*, Vol. 2(2): pp. 125-149, 1987.

[21] Colin C. Modélisation déclarative de scènes à base de polyèdres élémentaires. Thèse de Doctorat, Nantes, 1990.

[22] Colin C., Desmontils E., Martin J. and Mounier J.P. Working Modes with a Declarative Modeler. In *proceedings of the Compugraphics'97*, Portugal, December 1997.

[23] Davis E., Constraint Propagation with Interval Labels. In *Artificial Intelligence*, Vol. 32, 1987.

[24] Donikian S. and Hégron G. A Declarative Design Method for 3D Scene Sketch Modeling. In *Proceedings of Eurographics'93*, Vol. 12(33), 1993.

[25] Duisberg R. Animation Using Temporal Constraints: An Overview of the Animus System. In *Human-Computer Interaction*, Vol. 3: pp. 275-308, 1987.

[26] Elliot C., S. Greg, Yeung R., and Abi-Ezzi S. TBAG: A High level Framework for Interactive Animated 3D Graphics Applications. In *Proceedings of SIGGGRAPH'94*, pp. 421-434, Orlando, Florida, July 1994.

[27] Fa M., Fernando T. and Dew P.M. Direct 3D manipulation Techniques for interactive Constraint-based Solid Modelling. In *Proceedings of Eurographics'93*, pp. 237-248, 1993.

[28] Faltings B. Constraint Satisfaction Problem in CAD Systems. In *International Workshop on Constraint Processing in Computer Aided Design*, Swiss Federal Institute of Technology, Laussane, Switzerland, August 1994.

[29] Fowler B. Geometric manipulation of tensor-product surfaces. In *Proceedings of Interactive Workshop*, 1992.

[30] Freeman-Benson B., Maloney J., and Borning A. An Incremental Constraint Solver. In *Communications of the ACM*, Vol. 33 (1): pp. 54-63, January 1990.

[31] Freeman-Benson B. A module Compiler for ThingLab. In *Proceedings of the ACM Conference on Object Oriented Programming Systems, Languages and Applications*, pp. 389-396, New Orleans, October 1989.

[32] Freeman-Benson B. Converting an Existing User Interface to Use Constraints. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 207-215, Atlanta, Georgia, November 1993.

[33] Gleicher M. and Witkin A. Differential Manipulation. In *Graphics Interface'91*, pp. 61-67, June 1991.

[34] Gleicher M. A Graphics Toolkit Based on Differential Constraints. In *proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 109-120, Atlanta, Georgia, November 1993.

[35] Gobbeti E. and Balaguer J-F. VB2: An Architecture for Interaction in Synthetic Worlds. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 167-178, Atlanta, Georgia, November 1993.

[36] Gosling J. Algebraic Constraints. Ph.D. thesis, Carnegie-Mellon University, Published as CMU Computer Science Department Technical Report CMU-CS-83-132, May 1983

[37] Graf W.H. The constraint-based layout framework LayLab and its applications. In *proceedings of the Workshop on Effective Abstractions in Multimedia Layout, Presentations, and Interaction in conjunction with ACM Multimedia 95*, San Francisco, 1995.

[38] Haroud D. and Faltings B. Global Consistency for Continuous Constraints. Lecture notes in computer science 874: principles and practice of constraint programming, Alan Borning Editor, Springer Verlag, 1994.

[39] Heydon A. and Nelson G. The Juno-2 Constraints-Based Drawing Editor. SRC Research 131a, Systems Research Center, Digital Equipment Corporation, Palo Alto, California, December 1994.

[40] Hill R. D. A 2D Graphics Systems for Multi-User Interactive Graphics Based on Objects and Constraints. In *Advances in Object Oriented Graphics* I, Springer, Verlag, Berlin, pp. 67-91, 1990.

[41] Hill R. D., Brinck T., Rohall S., Patterson J.F., and Wilner W. The Rendezvous Architecture and Language for Constructing Multi-User Applications. In *Proceedings of ACM Transactions on Computer-Human Interaction*, Vol. 1(2), 1994.

[42] Horne B. Properties of User Interface Systems and the Siri Programming Language. In B. Myers, editor, Languages for Developing User Interface, pp. 211-236. Jones and Barlett, Boston, MA, 1992.

[43] Hsu C., G. Alt, Huang Z., Beier E., and Brüderlin B.D.. A Constraint-Based Manipulator Toolset for Editing 3D Objects. In *Proceedings of Fourth Symposium on Solid Modeling and Applications*, Atlanta, Georgia, 1997.

[44] Hudson S. Incremental Attribute Evaluation. A Flexible Algorithm for Lazy Update. In *ACM Transaction on Programming Languages and Systems*, Vol. 13: pp. 315-341, July 1991.

[45] Hyvönen E. Constraint reasoning based on interval arithmetic: the tolerance propagation approach. In *Artificial Intelligence*, Vol. 58, 1992.

[46] Kondo K. PIGMOD: parametric and interactive geometric modeller for mechanical design. In *Computer Aided Design*, Vol. 22(10): pp. 633-644, 1990.

[47] Kondo K. Algebraic method for manipulation of dimensional relationships in geometric models. In *Computer Aided Design*, Vol. 24(3): pp. 141-147, 1992.

[48] Kwaiter G., Gaildrat V. and Caubet R. Interactive Constraint System for Solid Modeling Objects. In *Fourth Symposium on Solid Modeling and Applications*, Atlanta, Georgia, 1997.

[49] Kwaiter G., Gaildrat V. and Caubet R. ORANOS: Un Solveur Dynamique de Contraintes Hiérarchiques et Géométriques. In *MICAD'97, International Journal of CADCAM and Computer Graphics*. Vol. 12(1-2), 1997.

[50] Kwaiter G., Gaildrat V. and Caubet R. Dynamic and Hierarchical Constraints Solver with Continuous Variables. In *Proceedings of JFPLC'97*, Sixth French Conference on Logic Programming and Constraint Programming, pp. 1-11, Orleans, 26-28 May 1997.

[51] Kwaiter G., Gaildrat V. and Caubet R. DEM2OMS: A High Level Declarative Modeler For 3D Graphics Applications. In *Proceedings of CISST'97*. International Conference on Imaging Science, Systems, and Technology, Las Vegas, Nevada, June 30-3 July 1997.

[52] Kwaiter G., Gaildrat V. and Caubet R. Integrating an Incremental Constraints Solver With a High Declarative Modeler for 3D Graphics Applications. In *Proceedings of PAP-PACT'98*. The Fourth International Conference and Exhibition on The Practical Application of Constraint Technology, London, UK, 23-27 March 1998.

[53] Kwaiter G., Gaildrat V. and Caubet R. Controlling Object Natural Behaviors in a 3D Declarative Modeler. In *Proceedings of CGI'98*, COMPUTER GRAPHICS INTERNATIONAL, Hannover Germany, June 22nd-26th, 1998. Published also in IEEE Computer Society Press.

[54] Leler W. Constraint Programming Languages: Their specification and generation. Addison-Wesley, Reading, MA, 1988.

[55] Lhomme O. Consistency Techniques for Numeric CSPs. In *Proceeding of the 13th International Joint Conference on IA*, pp. 232-238, 1993.

[56] Liege S. and Hégron G. An Incremental Declarative Modelling applied to Urban Layout Design. In *Proceedings of WSCG'97*. Vol. 2, pp. 272-281, February 1997.

[57] Light R. and Gossard D. Modification of geometric models through variational geometry. In *Computer Aided Design*, Vol. 14(4): pp. 209-214, July 1982.

[58] Loranzo-Pérez Tomas. Spatial Planning. A configuration Space Approach. In *IEEE Transactions on Computers*, Vol. 32 (2): pp. 108-120, February 1983.

[59] Mackworth A. K. Constraint Satisfaction. In Stuart C. Shapiro, Editor, Encyclopedia of Artificial

Intelligence, Vol. 1: pp. 285-293, Second edition, John Wiley and Sons, 1992.

[60] Maloney J. Using Constraints for user Interface Construction. Ph.D. thesis, Dep. of Computer Science and Engineering, University of Washington, August 1991.

[61] Martin P., Martin D. Declarative generation of a family of polyhedra. In *Proceedings of Graphicon'93*, September 1993.

[62] McDonald, J. Stuetzle A.Werner, and Anderas B. Painting Multiple Views of Complex Objects. In *Proceedings of the 1990 ACM Conference on Object Oriented Programming: Systems, Languages and Applications and The European Conference on Object Oriented Programming*, Ottawa, Canada, October, pp. 245-257, 1990.

[63] Miyashita K., S. Matsuoka, S. Takahashi, A. Yonezawa, and T. Kamada. Declarative programming of graphical interfaces by visual examples. In *Proceedings of the UIST'92, ACM SIGGRAPH Symposium, on User Interface Software and Technology*, pp. 107-116, Monterey, Canada, 1992.

[64] Myers Brad A., Dario A. Guise, Roger B. Dannenberg, Vander Zanden B., Kosbie D., Pervin Ed., Mickish A., and Marchal P. Garnet: Comprehensive Support for Graphical, Highly-Interactive User Interfaces. In *IEEE Computer*, Vol. 23: pp. 71-85, November 1990.

[65] Nelson G. Juno, A Constraint Based Graphics System. In *Proceedings of. SIGGRAPH'85*: pp. 235-243, 1985.

[66] Olsen, Jr., Dan R. Creating Interactive Techniques by Symbolically Solving Geometric Constraints. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, Snowbird, Utah, pp. 102-107, 1990.

[67] Owen J. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the ACM Symposium of Solid Modeling and CAD/CAM Applications*, pp. 397- 407, 1991.

[68] Plemenos D., Tamine K. Increasing the efficiency of declarative modeling. Constraint evaluation for the hierarchical decomposition approach. In *Proceedings of* WSCG'97. Vol. 2, February 1997.

[69] Prosser P. Hybrid alghorithms for the constraint satisfaction problem. In *Computational Intelligence*, Vol. 22:268-299, 1993.

[70] Poulet F., Lucas M. Modeling Megalitic sites. In *Proceedings of Eurographics'96*, September 1996.

[71] Rossignac J. Constraints in Constructive Solid Geometry, In *Proceedings of ACM Workshop on Interactive 3D Graphics*, pp. 93-110, Chapel Hill, 1986.

[72] Sannella Michael. SkyBlue: A multi-Way Local Propagation Constraint Solver for User Interface Construction. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, Marina Del Rey, California, November 1994.

[73] Sistare Steven. A Graphical Editor for Constraint-Based Geometric Modeling. Ph.D. thesis, Department of Computer Science, Harvard, Published as Technical Report TR-06-91, December 1990.

[74] Sunde G. Specification of shape by dimensions and other geometric constraints. In *Proceedings of IFIP WG.5.2on Geometric Modeling*, Rensselaerville, NY, Mai 1986.

[75] Sunde G. A CAD system with declarative specification of shape. In *Eurographics Workshop on Intelligent CAD Systems*, Noordwijkerhout, April 1987

[76] Sussman, G. J., and Steel, G. L. CONSTRAINTS A language for expressing almost hierarchical descriptions. Artificial Intelligence, Vol. 14 (1): pp. 1-39, 1980.

[77] Sutherland I. Sketchpad: A Man-Machine Graphical Communication System. In *proceedings of the Spring Joint Computer*, pp. 329-346, 1963.

[78] Van WYK, C. J. A high-level language for specifying pictures. In *proceedings of the ACM Transaction Graphic*, Vol. 1 (2): pp. 163-182, 1982.

[79] Vander Zanden Brad. An Incremental Planning Algorithm for Ordering Equations in Multilinear system of Constraint. Ph.D. thesis, Department of Computer Science, Cornell University, April 1988.

[80] Vander Zanden Brad: An Incremental Algorithm for Satisfying Hierarchies of Multiway Dataflow Constraints. In *proceedings of the ACM Transactions on Programming Languages and Systems*, Vol. 18 (1): pp. 30-72, January 1996.

[81] Waltz D. Understanding line drawings of scenes with shadows. The Psychology of Computer Vision, 1987.

[82] Welch W., Gleicher M., Witkin A. Manipulating surfaces differentially. In *Proceedings Compugraphics'91*, September 1991. Also appears as CMU School of Computer Science Technical Report CMU-CS-91-175.

[83] Welch W. and Witkin A. Variational surface modeling. In *Proceedings of Computer Graphics*, Vol. 26, 1992.

[84] Wilk M.R. Equate: An object-oriented constraint solver. In A. Paepcke, editor, In *proceedings of the* OOPSLA'91, *ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 286-298, New York, NY, October 1991.

[85] Wu Wen-Tsün. Basic principles of mechanical thorrem proving in geometries. In Journal of Systems Sciences and Mathematical Sciences, Vol. 4: pp. 207-235, 1986.