Automatic Reasoning for Design Under Geometric Constraints

M. Shpitalni, H. Lipson
Laboratory for Computer Graphics and CAD
Faculty of Mechanical Engineering
Technion, Haifa, Israel 32000
e-mail: shefi@tx.technion.ac.il

Abstract

Parametric design is very stable but requires a predefined dimensioning and ordering scheme, thus limiting flexibility and precluding sketch input. Variational geometry design, while general and flexible, necessitates intensive use of numerical solvers to solve many simultaneous nonlinear equations. Frequently the solvers cannot solve these equations. A new system, based on an original theory for automatic constraint analysis, has been developed for solving sets of two-dimensional geometric constraints in product design. The proposed system offers the flexibility of variational based design along with the stability of parametric design. The solution strategy is based upon breaking down the problem into a sequence of construction steps. When no sequential construction is found, auxiliary geometrical constructions are automatically generated based on rules for relocating constraints. Thus, an apparently simultaneous constraint set is converted into a set that can be constructed sequentially by decomposing strongly connected components of the original constraint graph. This new approach has been implemented in a system for designing sheet metal parts.

Keywords: CAD, Conceptual Design, Geometry Constraint

1. Introduction

Two design paradigms are used and explored for design and manufacturing applications: *feature-based* design and *constraint-based* design [3]. In constraint-based design, shapes are specified by means of geometrical constraints that relate shape features to shape parameters. The constraints are typically specified based on predefined topological arrangements of features (sometimes entered by means of a sketch) to provide a context for the problem. Studies of constraint solution range from pure geometry, e.g., [1], to kinematics [6], engineering constraints [5], and theorem proving [2].

When solving geometrical constraint problems, a solver must provide an instance of the given topology that *exactly* satisfies the given constraints. Two major approaches can be identified: (1) *parametric* geometry, in which constraints are given so that the desired shape can be constructed sequentially according to a predefined scheme and order, and (2) *variational* geometry, where constraints are given by an arbitrary scheme in no particular order. The solver must then derive a solution strategy automatically in order to construct the desired shape.

Since the parametric approach uses a predefined scheme of dimensions and a predefined evaluation order, the solution process is more stable and controllable, and is therefore more common in commercial CAD systems. However, the need to adhere to a prespecified dimensioning scheme and order limits the freedom of the designer to modify the definition of the shape. Moreover, the desire to allow more flexible input methods for sketching [8] and conceptual design necessitates that dimensioning of the design be independent of a specific dimensioning scheme and order. However, lack of a solution strategy makes variational geometry problems more difficult to solve because the solver must be capable of the following: deriving a solution sequence; handling large sets of simultaneous non-linear equations; managing multiple solutions; and determining user intent as the most plausible solution.

Variational geometry problems are usually solved by using either *numerical solvers* or *sequential construction solvers* based on applying precoded steps, which may resort to numerical methods in special cases. In essence, numeric solvers are more general but construction solvers are more stable. This paper describes a novel solution technique which allows solving more complex problems using construction solvers without resorting to numerical methods. The new solver is based on automatic generation of auxiliary geometrical constructions and constraints. While the additional constraints do not actually add any new mathematical information, they do simplify the automatic search for a solution strategy as well as significantly enhance the analytical capabilities of the sequential solver. The new solver has been implemented successfully in a CAD system [10] for design of sheet metal parts.

This paper first describes existing techniques for solving geometrical constraint problems and places the proposed method in context. Then, the new method is fully described and demonstrated through examples and implementation.

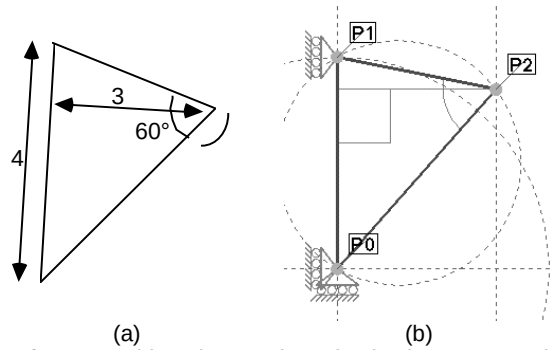
2. Geometrical constraint solvers

Variational geometrical constraints are usually specified based on an initial sketch that reveals the topological relationships among the geometrical entities. Constraint problems are traditionally solved using a *numeric* approach, whereby geometrical constraints are translated into general algebraic expressions solved using iterative methods, such as Newton-Raphson or homotopy [7]. However, despite their generality, numerical solvers often prove to be inadequate for large sets of non-linear equations and rely to a great extent on the quality of the initial sketch. Numeric solvers are also unable to explore the solution space and provide almost no information upon failure. The alternative of solving these equations using symbolic algebra does not appear to be practical [9]. Consequently, a different approach has been adopted, based on *constructive constraint solvers* used for satisfying constraints using a constructive sequence of steps. This process resembles

building a shape with a compass and protractor [1,4].

Figure 1(a) illustrates a simple geometrical constraint problem in which a triangle is sought under three constraints specifying width, height and spanning angle. Although a numerical system consisting of three nonlinear equations can be formulated and solved (two distances and one angle constraint), the problem can also be solved using three simple construction steps. Figure 1 (b) shows the solution generated automatically by a constructive solver.

The problem illustrated in Fig. 1 is considered simple because it can be solved by executing a sequence of construction steps using the given constraints only, applied one at a time. This capability is referred to as the basic sequential construction solver, shown in Fig. 2. This simple solver relies on executing step-by-step operations on a case-specific basis. An operation is pre-coded for each constraint type; for example a circular locus is pre-coded for a point constrained by distance from a known location. Loci intersections make it possible to derive the exact point location or possible set of locations. Table 1 lists the solution transcript for solving the above problem. Note how the automated solution process associates point loci with constraints and then intersects these loci when possible. It also uses three additional ground constraints that may be supplied by the user or temporarily added by the system, automatically and without loss of generality.



(a) (b)

Fig 1. A problem that can be solved using a sequential process: (a) initial problem; (b) solution.

Contrary to numerical solvers, a simple constructive solver is completely stable and controllable. Precoded formulations make it possible to derive solutions efficiently while handling all possible singularities and explicitly considering alternative solutions. Therefore, the process is also less dependent on the initial sketch. The apparent drawback of this approach is that only precoded constraints can be handled. This drawback is less conspicuous in light of the fact that the repertoire of supported constraints is in any case limited by the user interface of modern CAD systems. However, the real restriction on the basic solver is that the constraints of the given problem do not always permit a direct sequential step-by-step solution. In such cases, some solvers try using more deductive rules or resort to a numeric solution, as illustrated in Fig. 2.

Graph constructive solvers use a graph representing the constraint dependencies to analyze and find a valid sequence to the solution using the given constraints, as in [1]. Several techniques for performing this analysis were suggested, some called *constraint propagation* methods [11]. Such an analysis can yield a sequential path to the solution if one exists, or allow identification of isolated problem components that can be solved independently and then viewed as a single *substructure*. When the given constraints do not facilitate finding a path or substructuring, *rule constructive* solvers use rewrite rules to discover new steps.

Given:
C0:P0.X=6.0 C3:D(P2,(P0;P1))=-3.00
C1:P0.Y=8.0 C4:D(P0,P1)=4.00
C2:P1.X=6.0 C5:A(P1,P2,P0)=60`

Solving:

Using ground-x constraint C0:
Locus P0L0: Point P0 is on line 1.00x+0.00y+-6.00=0
Using ground-y constraint C1:
Locus P0L1: Point P0 is on line 0.00x+1.00y+-8.00=0
Combining locus P0L0 and P0L1, point P0 is at 6.00;8.00
Using ground-x constraint C2:
Locus P1L0: Point P1 is on line 1.00x+0.00y+-6.00=0
Using point-point distance constraint C4 and point P0:
Locus P1L1: Point P1 is on circle at 6.00;8.00 with R=4.00
Combining locus P1L0 and P1L1, point P1 is at 6.00;12.00 or 6.00;4.00 (Selected 1st)
Using angle constraint C5 and pointsP1, P0:
Locus P2L0: Point P2 is on circle at 7.15;10.00 with R=2.31
Using point-line distance constraint C3 and points P0, P1:
Locus P2L1: Point P2 is on line 1.00x+0.00y+-9.00=0
Combining locus P2L0 and P2L1, point P2 is at 9.00;11.39 or 9.00;8.61 (Selected 1st)

Table 1. Transcript for problem in Fig. 1. Note that no additional constraints were derived.

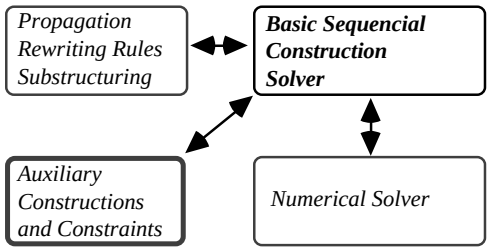


Fig 2: The variational constraint solver system: Basic Sequential Construction Solver and its related components. The proposed technique is encapsulated in the Auxiliary Construction Component.

3. Constraint generator using rewriting rules

The simplest approach to unlocking dead-end situations in the basic sequential construction solver is to generate additional constraints dependent on given ones that have the potential to allow the solver to proceed. Constraints are derived by standard geometrical rules. Our solver uses the following three well known rules:

Accumulation rule: If the angle between lines *A* and *B* is known or constrained to α and the angle between lines *B* and *C* is known or constrained to β, then the angle between *A* and *C* is constrained to $\alpha+\beta$. To implement this rule, angles must be manipulated with their proper arithmetic signs denoting clockwise and counter-clockwise directions. Similarly, the accumulation rule can be applied to distance ratios. The accumulation rule automatically accounts for higher rules, such as: (a) the angles around a polygon amount to *180*(n-2)* degrees where *n* is the number of vertices; (b) if line *A* is perpendicular to line *B* and line *B* is parallel to line *C*, then line *C* is also perpendicular to line *A*; etc.

Law of sines and law of cosines between the angles and sides of a triangle.

Consider, for example, the variational geometry problem illustrated in Fig. 3. In contrast to to the previous example, this problem cannot be solved sequentially by applying only the *given* constraints one at a time. However, applying the above rules will make a solution

possible.

The transcript of the solution process for the above problem, excluded due to space restrictions, indicates that the cosine, sine and angle accumulation laws have been used.

By using these rules and re-analyzing the constraint graph to find paths to the solution or to substructures, the constructive solver can to handle a wider variety of problems. Nevertheless, faced by constraints requiring simultaneous solution, the solver may exhaust all possibilities for progressing. In such cases, a method of auxiliary constructions or *constraint relocation* is proposed.

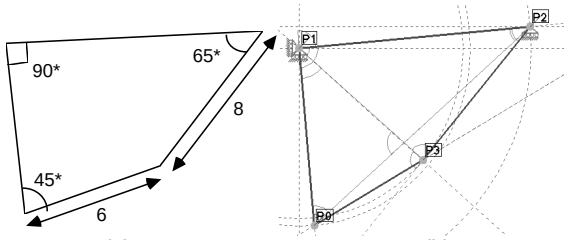


Fig 3: Using rewriting rules: (a) Initial constraint problem; (b) solution.

4. **Auxiliary construction and constraint relocation**

The auxiliary construction and constraint relocation approach involves *relocating* existing constraints to a more effective location on the constraint graph, thus permitting sequential advance and breaking up strongly connected components of the constraint graph so they can be resolved sequentially.

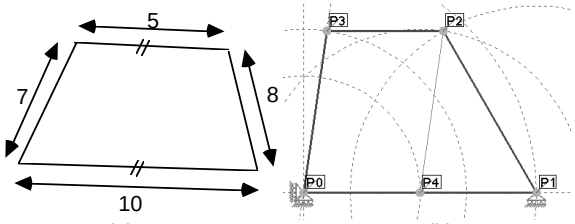


Fig 4. Constraint relocation: (a) Initial problem; (b) solution. Note the automatic generation of point 4.

First, consider an auxiliary construction used to solve the problem illustrated in Fig. 4. The given constraints cannot be applied sequentially to yield a solution, nor can the constraints provided by the above rewriting rules. However, construction of an auxiliary line (**p2-p4**) parallel to the left side of the trapezoid makes the problem solvable using sequential steps only. The solution process transcript is presented in Table 2.

Refer to Figure 5. The constraint relocation process is described here for a simple case of three points in two dimensions and a single-step relocation. Consider a set of three points as follows: point **p1** is known, and points **p2** and **p3** are unknown. In addition, there is a constraint c1 between points **p1** and **p2**, and two constraints, c2 and c3, between **p2** and **p3**. Theoretically, this is a sequential dead-end: point **p2** cannot be determined because there is only one constraint relating it to a known point. Similarly, point **p3** cannot be determined because **p2** is unknown, even though point **p3** is well constrained to **p2**. In fact, **p2** together with **p3** constitute a substructure which is fully resolved internally. For certain constraint type configurations, the following procedure will allow relocating constraint c1 (which currently constrains **p2** to the known point **p1**) to **p3,** thus constraining **p3** to a known point. The addition of a constraint for **p3** may improve the sovability of **p3** or otherwise revitalize the constraint pool.

Given:
C0:P0.X=6.0 *C4:D(P2,P3)=5.00*
C1:P0.Y=7.0 *C5:A(P2,P3,P0,P1)=0'*
C2:P1.Y=7.0 *C6:D(P1,P2)=8.00*
C3:D(P0,P1)=10.00 *C7:D(P0,P3)=7.00*
Solving:
Using ground-x constraint C0:
Locus P0L0: Point P0 is on line 1.00x+0.00y+-6.00=0
Using ground-y constraint C1:
Locus P0L1: Point P0 is on line 0.00x+1.00y+-7.00=0
Combining locus P0L0 and P0L1, point P0 is at 6.00;7.00
Using ground-y constraint C2:
Locus P1L0: Point P1 is on line 0.00x+1.00y+-7.00=0
Using point-point distance constraint C3 and point P0:
Locus P1L1: Point P1 is on circle at 6.00;7.00 with R=10.00
Combining locus P1L0 and P1L1, point P1 is at 16.00;7.00 or -4.00;7.00 (Selected 1st)
Using point-point distance constraint C6 and point P1:
Locus P2L0: Point P2 is on circle at 16.00;7.00 with R=8.00
Using point-point distance constraint C7 and point P0:
Locus P3L0: Point P3 is on circle at 6.00;7.00 with R=7.00
Translocating constraints using the parallelogram principle
Added point P4 at (roughly) 10.90;7.30
Translocating C7 into C8:D(P2,P4)=7.00
Translocating C4 into C9:D(P0,P4)=5.00
From parallelogram principle:
C11:A(P2,P3,P0,P4)=0' C10:A(P0,P3,P2,P4)=0'
Using point-point distance constraint C9 and point P0:
Locus P4L0: Point P4 is on circle at 6.00;7.00 with R=5.00
Deriving from C5, C11: C12:A(P1,P0,P4)=0`
Using angle constraint C12 and points P0, P1, P0: Locus P4L1: Point P4 is on line 0.00x+1.00y+-7.00=0
Combining locus P4L0 and P4L1, point P4 is at 11.00;7.00 or 1.00;7.00 (Selected 1st)
Using point-point distance constraint C8 and point P4:
Locus P2L1: Point P2 is on circle at 11.00;7.00 with R=7.00
Combining locus P2L0 and P2L1, point P2 is at 12.00;0.07 or 12.00;13.93 (Selected 2nd)
Using angle constraint C10 and points P3, P2, P4:
Locus P3L1: Point P3 is on line 0.99x+-0.14y+-4.94=0
Combining locus P3L0 and P3L1, point P3 is at 7.00;13.93 or 5.00;0.07 (Selected 1st)

Table 2: Transcript of solution for Fig. 4. Note the use of constraint relocation.

First, an auxiliary point **p4** is constructed, and relative constraints c2 and c3 relating **p2** and **p3** are copied as c2' and c3' relating **p1** to the new point **p4**. At the same time, constraint c1 relating **p1** and **p2** is copied as c1' between **p3** and **p4**. The second step is allowed only for certain configurations of constraints (c1, c2, c3) that retain their validity across this relocation. Now, **p4** can be derived since it is constrained by two constraints to the known point **p1**, and point **p3** is now constrained by c1' to the known point **p4**.

This principle can be applied, for example, when constraints c1, c2 and c3 are distance, distance ratio and angle constraints, thus generating a parallelogram (**p1,p2,p3,p4**), as in Fig. 4. Figure 6 demonstrates a second application of the parallelogram, where the relocated constraint is a distance-ratio constraint. In that sequence, two relocations must take place before the problem can be solved sequentially.

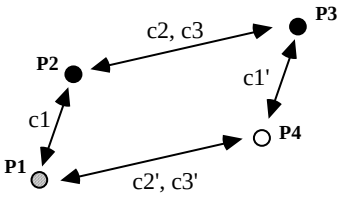


Fig 5: The constraint relocation sequence.

5. Implementation

A two dimensional sequential construction solver has been implemented using the above techniques. The solver uses *points* as its basic elements. To completely solve a two-dimensional constraint problem pertaining to *n* points, exactly *2n* independent constraints must be supplied since there are *2n* degrees of freedom related to point positioning. This is a necessary but not sufficient condition since the given constraints may specify impossible geometry or may not adequately cover all the points. Of these *2n* constraints, *3* constraints must be absolute to fix the rigid body in space, while the rest may be relative. If only shape structure is required and not absolute position, then only *2n-3* independent constraints are necessary, and the remaining *3* ground constraints can be added anywhere as convenient.

The following two-dimensional constraints types are supported:

- Point-to-point distance (among two points)
- Point-to-line distance (among three points)
- Angle between two lines (among four/three points)
- Distance/length ratio (among four/three points).
- Ground *x* or *y* coordinate (constraining one point).

These five constraints are sufficient to specify most two-dimensional constraint problems involving points, lines and arcs produced by standard CAD/CAM systems.

The solver has been implemented as part of a CAD system for designing and manufacturing sheet metal products [10].

The solver accepts a two-dimensional freehand sketch, shown in Fig. 6(a), that was acquired and transformed into a two-dimensional graph. The user may explicitly place constraints, or the system may try to infer these constraints automatically according to geometrical regularities evident in the sketch. Regardless of their source, the constraints are reformulated as constraints between points, as shown in Fig. 6(b). The geometry is then grounded and resolved using the above methods to yield the solution shown in Fig. 6(c). The final product can then be manufactured, as illustrated in Fig. 6(d).

11. Conclusions

A constructive geometrical constraint solver has been described which provides a stable and controllable means for solving problems consisting of a fixed repertoire of constraint types. The solver incorporates a new technique for relocating constraints using auxiliary constructions that allow the solver to proceed using sequential steps without resorting to numerical solutions, thus retaining solver stability.

We intend to further develop the constraint relocation approach using more auxiliary constructions such as the parallelogram structure. We also intend to explore the possibilities of applying these principles to solving variational geometry problems in three dimensions.

Acknowledgment

This research has been supported in part by the Fund for the Promotion of Research at the Technion, Research No. 033-028.

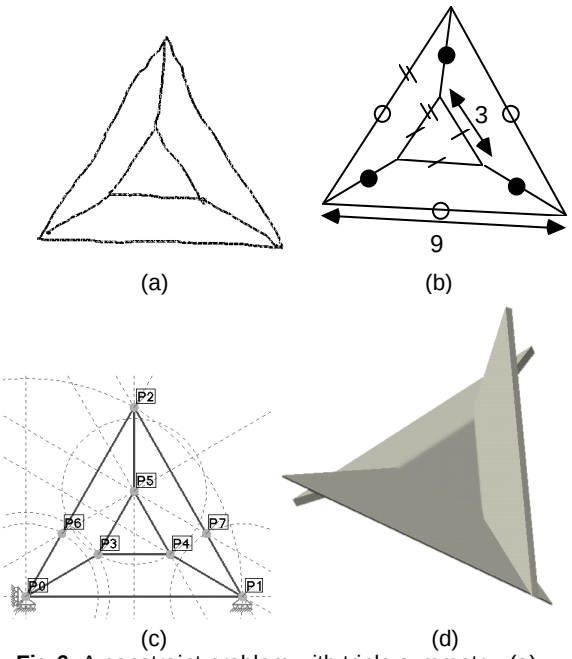


Fig 6: A constraint problem with triple symmetry: (a) Initial sketch; (b) schematic drawing of constraints; (c) solution; (d) image of product.

References

[1] Bouma W., Fudos I., Hoffmann C. M., Cai J., Paige R., 1995, Geometric constraint solver, Computer Aided Design, 27(6): 487-501.

[2] Chou C. S., 1987, *Mechanical Theorem Proving*, D. Reidel Publishing.

[3] Hoffmann C. M., Rossingnac, J. R., 1996, A roadmap to solid modeling, IEEE Transactions on visualization and computer graphics, 2(1): 3-10.

[4] Joan-Arinyo R., Soto A., 1996, A ruler-and-compass geometric constraint solver, 5th IFIP Workshop on Geometric Modeling in CAD, Airlie, VA, May 19-23, 1996.

[5] Kimura F., Suzuki H., Ando H., Sato T., Kinosada A., 1987, Variational geometry based on logical constraints and its applications to product modeling, Annals of the CIRP 36(1): 75-78.

[6] Kramer G., 1992, *Solving geometric constraint systems*, MIT Press, Cambridge, MA.

[7] Lamure H., Michelucci D., 1996, Solving geometric constraints by homotopy, IEEE Transactions on Visualization and Computer Graphics, 2(1): 28-34.

[8] Lipson H., Shpitalni M., 1995, A new interface for conceptual design based on object reconstruction from a single freehand sketch, Annals of the CIRP, 45(1): 133-136.

[9] Manocha D., 1994, Solving systems of polynomial Equations, IEEE Computer Graphics and Applications, 3: 46-55.

[10] Shpitalni, M., 1993, A new concept for design of sheet metal products, Annals of the CIRP, 42(1):123-126.

[11] Sridhar N., Agrawal R., Kinzerl G., 1996, Algorithms for the structural diagnosis and decomposition of sparse, underconstrained design systems, Computer Aided Design, 28(4): 237-249.