Extending a constructive solver with constraints inferred by variable topology parametrisation

Evgueni N. Loukipoudis
Department of Computer Science
University of Sofia, Bulgaria
5, James Bouchier blvd., 1126 Sofia
evgueni@fmi.uni-sofia.bg

Abstract

The paper presents the internal graph-constructive representation of parametrised geometry in a CAD system that supports the design of objects with variable topology. A constructive constraint solver is extended by adding two implicit constraints: *ed* (equal distance) and *ea* (equal angle). The necessity of supporting both constraints in a model with structural parameters is argumented. The basic solvable patterns involving these constraints and the forming and merging of circularly linked clusters is presented. The applicability of the constraint solving to hierarchically structured parametric objects is also shown with a practical example.

Keywords: Constraint-based design, Parametric models, Variational models

Introduction

Modelling by constraints has become a preferred method of defining geometry in CAD systems. This process begins with drawing a rough sketch and is followed by stepwise fixing of the degrees of freedom of the object's components. The latter is done adding geometric constraints such as tangency, perpendicularity, prescribed distance and angle between primitives, etc., usually depicted as annotations in engineering drawings. The CAD system is then attempting to satisfy all imposed constraints in a process referred to as *constraint solving.* The constraint model is attractive also for the ease with which complex modifications to the geometry can be carried out. Constraints can be removed and/or substituted, and families of parts with variable dimension values can be represented.

As far as geometric constraints are directly represented by algebraic equations, *numerical constraint solving* has been widely used in a number of systems [2], [5], [8], [12]. In addition to its generality, the method supports integral constraints such as prescribed area, perimeter, moment of inertia. However, the iterative solver needs an instance to start with and might converge to an undesired solution.

Another class of solvers are based on the constrictive approach. Geometric elements are placed in some order either following a set of construction rules – *rule-constructive solving* [1] – or building a graph of geometric elements linked by constraints and attempting to reduce it to a set of predefined solvable patterns – *graph-constructive solving* [3], [9]. The second method is more robust and may be guided to select the proper solution, but graph analysis is strongly dependent on the set of possible constraints.

Most constructive solvers deal with a restricted number of constraints: prescribed distance and angle, incidence and parallelism, and still their scope goes beyond the class of problems solvable by ruler, compass and protractor. However, they lack the support of some typical constraints representing the relations that exist between geometric elements of the so-called *parametric objects.*

In this paper we discuss the extension of a graph-constructive solver to handle the instantiation of objects that may in general have structural parameters. We concentrate on some new constraints required when modelling repetitive structure and the basic construction steps needed to support them.

Parametric models and constructive constraint solving

Parametric models have often been opposed to variational, constraint-based ones. The latter may represent a wide class of families of parts by utilising explicit constraints, such as prescribed dimensions involving parameters rather than fixed values. A *parametric object* is a more general concept being the description of a class of objects, the individual members of which (called *instances*) are uniquely identified by the values of a set of attributes, called *input parameters.*

Parametric objects with structural parameters

One of the main difference lies in the fixed topological structure of a constraint problem. Conversely, a parametric object may have *structural* parameters to describe similar but not identical topology between instances [7]. A typical example is a cog-wheel with a parametrised number of cogs or the comb-like polygon in Fig. 1 with a varying number of slots Ns.

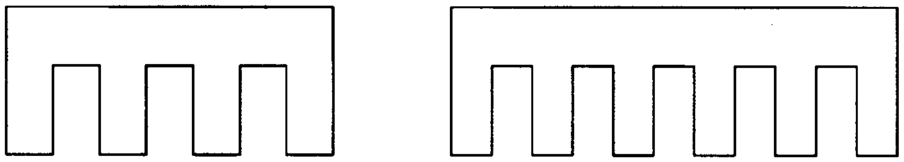

Figure 1: Two instances of a COMB with $Ns = 3$ and $Ns = 5$

Attempts have been made to introduce structural parameters in a constraint-based model, but they are based on performing modelling operations like *circular repetition* on a set of elements already constructed [10]. This implies that geometric elements can be split in groups of constructable sets which can after be replicated in a circular or rectangular manner. A similar idea has lead to the definition of the *circular* and the *matrix pattern features* in the Step standard. Representation in such a mixed procedural-declarative manner might not always be possible, which is illustrated with the same object shown in Fig. 1. Editable models are more suitable [6], [11], but they require an internal representation involving a larger number of constraint types than usual.

Constraints representing equal dimensions

A typical implicit constraint inferred by parametrising topology is the *equal distance.* It represents the requirement that the distance between two points should be equal to the distance between another pair of points. A similar implicit constraint *equal angle* relates two pairs of lines, so that the angle between two of them is equal to the angle between the other two.

Such constraints are produced when defining that a set of geometric elements are equally spaced which is the case with the object in Fig. 1. In a parametric model the actual distance between the slots will not be specified, and the *ed*-constraint is a natural way of representing the fact that they are equally spaced.

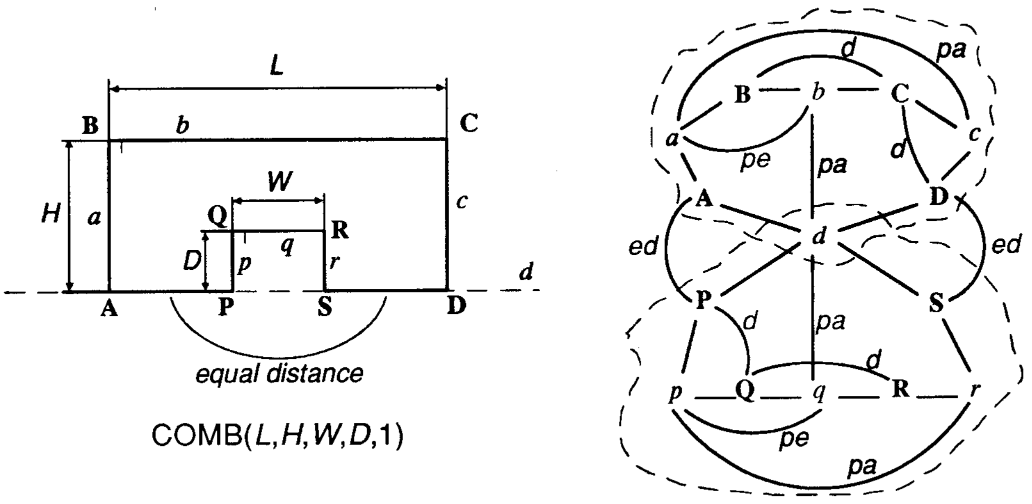

Figure 2: An instance of COMB with $Ns = 1$ and the corresponding constraint graph

Figures 2 and 3 show two instances of the object COMB and the graphs, representing the relations between the constituting elements. The nodes of the graph are the points and the lines (the construction lines rather than the segments that connect the points) of the object and they are linked by arcs representing the imposed constraints. We have marked the arcs of the graph with the type of the constraint as follows: **pa** – parallelism; **pe** – perpendicularity; **d** – prescribed distance. By arc with no label we denote the incidence constraint.

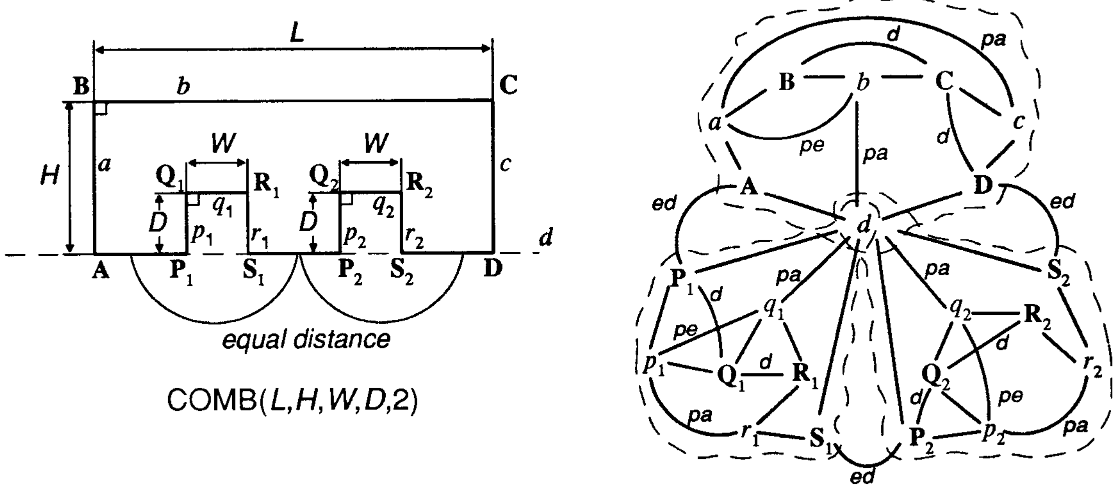

Figure 3: An instance of COMB with $Ns = 2$ and the corresponding constraint graph

Equal angle constraints would be used for objects that possess circular repetitive structure to denote the equal angular dimension between the adjacent items. This constraint would be expressed as the natural language specification: *A cog-wheel with a variable number of cogs, being located at equal angles around a common centre.*

The graph-constructive solving

We shall first briefly present the process of graph reduction, its analysis and the placing of geometric elements, which does not differ in its central part from the process described in [3] and [9]. The model is first translated into an internal representation with fixed topology. This is the graph whose vertices are the characteristic points and construction lines and circles rather than the arcs and line segments comprising the drawing.

We use a pure bottom-up approach. The graph is first decomposed to a set of clusters built up by elements for which the execution of a construction step is possible. The cluster formations are then analysed in a recursive manner to discover solvable patterns treating clusters as single geometric elements. Clusters are grouped and placed with respect to each other until no more isolated clusters (elements) exist.

The algorithm is made of two steps:

1. FORM A NEW CLUSTER: Select a pair of geometric elements, related by a constraint (excluding *ed*- or *ea*-constraint), so that a maximum number of constraints relates this pair to other graph vertices. Make a new cluster comprising these two elements. This aims at forming a cluster that consists of as many elements as possible.
2. JOIN TO A CLUSTER: Select a cluster and for each element (or another cluster) not belonging to the cluster that has two dimensional or implicit constraints relating it to elements already in the cluster execute a construction step. The element (or cluster) can be located wrt the selected cluster and thus become a new cluster member.

The second step is actually the core of the construction process. A set of basic patterns for locating an element related by dimensional or implicit constraints to two other elements with known position is defined as the solution of a simple system of equations. This solution gives the parameters of a translation and rotation that is applied to the newly added element. Examples of such patterns involving dimensional (explicit) constraints are the basic steps:

- Put a point at prescribed distances from two other points;
- Put a point at prescribed distances from a point and a line;
- Put a line at prescribed distance from a point and at an angle from another line;
- Put a line at prescribed angles with two other lines; etc.

Note that the process ends when all geometric elements have been merged in a single cluster, i.e. no isolated elements and/or clusters exist. This single cluster possesses the three additional degrees of freedom on the plane.

Apart from the problem with the exponential number of solutions some precautions must be taken when dealing with the *parallelism* constraint [4]. The reason for it is that two parallel lines, that are not related by an additional distance constraint, do not form a rigid cluster, i.e. the latter is not relocatable by applying translation and rotation only.

One of the methods of extending a constructive constraint solver is to form small systems of equation inferred by patterns of two and more constraints. An example of placing two clusters linked by three different constraints is to satisfy two of them and derive the loci of the element involved in the third constraint. Although very powerful this method is hard to apply for the constraints in question as they are not explicit, and can not be isolated and solved separately.

Instead, we propose to add a number of construction steps corresponding to a set of solvable configurations of clusters linked by *ed*- or *ea*-constraints. We should not account for all possible cases, but rather concentrate on the ones usually generated by varying topology parametrisation.

Solving the additional implicit constraints

We shall first present the basic construction steps taken to merge two clusters linked by an *ed*-constraint and sharing a common element. The most important is the configuration when the constraints are imposed on points (rather than lines), which is shown in detail. The specifics of *ed*-constraints linking a point and a line and two lines are discussed briefly as well.

Two *ed*-clusters sharing a common line

Figure 4 shows the general case of two *ed*-clusters α and β sharing a common line l. The relative position of each pair of points in the cluster they belong is fixed, and has been determined during cluster construction. The next step is performed by fixing one of the pairs and letting the other slide along the line keeping $\mathbf{C}$ and $\mathbf{D}$ on a constant distance from it until the condition $\|\mathbf{A}-\mathbf{C}\| = \|\mathbf{B}-\mathbf{D}\|$ is satisfied. Obviously, the motion of the second cluster is translational. In the trivial case all four points lie on the shared line as shown in Fig. 2. Note also that the two cluster may share more than one line if these lines are parallel.

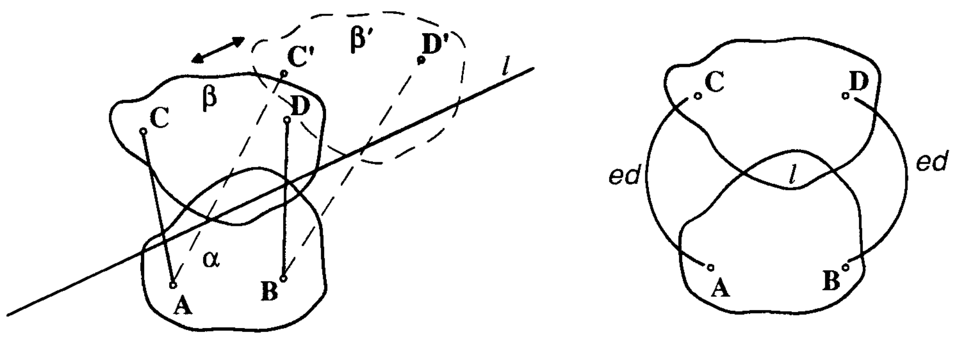

Figure 4: Cluster β may slide along the line

Let the line has coordinates $(\mathbf{n}^\alpha, c^\alpha)$ in cluster α and $\left(\mathbf{n}^\beta, c^\beta\right)$ in cluster β. To form the equation to solve we need to rotate both clusters so that the common line becomes parallel to the x-axis and then translate vertically the clusters to coincide with the x-axis. Cluster β has one degree of freedom which is the unknown u as the x-component of the translation. Thus, the two transformation matrices in homogenous coordinates are:

$$\mathbf{M}_\alpha = \begin{bmatrix} n_x^\alpha & n_y^\alpha & 0 \\ -n_y^\alpha & n_x^\alpha & -c^\alpha \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{M}_\beta(u) = \begin{bmatrix} n_x^\beta & n_y^\beta & u \\ -n_y^\beta & n_x^\beta & -c^\beta \\ 0 & 0 & 1 \end{bmatrix}.$$

The equation to solve is then:

$$\|\mathbf{M}_\alpha \cdot \mathbf{A} - \mathbf{M}_\beta(u) \cdot \mathbf{C}\| = \|\mathbf{M}_\alpha \cdot \mathbf{B} - \mathbf{M}_\beta(u) \cdot \mathbf{D}\|,$$

which is linear wrt to the unknown u due to its symmetric position at both sides. Thus, in case of imposing orientation on the construction lines the uniqueness of the solution at this step is guaranteed. The actual distance between the points in the pairs can then be computed, but this is not necessary as the clusters can be merged directly by transforming all elements of β by the matrix

$$\mathbf{M}_{\beta\alpha} = \mathbf{M}_\alpha \cdot \mathbf{M}_\beta(u).$$

Clearly, an equal distance constraint can be either redundant or contradictory, i.e. the equation may have infinite or zero number of solutions. The uniqueness is only possible if:

$$\mathbf{n}^\alpha \cdot (\mathbf{A}-\mathbf{B}) \neq \mathbf{n}^\beta \cdot (\mathbf{C}-\mathbf{D}).$$

It seems easy to extend the scope of the solver by generalising the *ed*-constraint involving simple functions of the distance. For example, one might wish to specify that the distance between a pair of points must be twice as big as the distance between another pair of points. This will lead to the cluster formation shown in Fig. 5 and to a slight modification to the equation we derived above. Unfortunately, this equation is no longer linear and two possible solutions exist. Both solutions of a simple configuration when all the points lie on the shared line and the relation between the constraints is given by $f(t) = t/2$ is also shown.

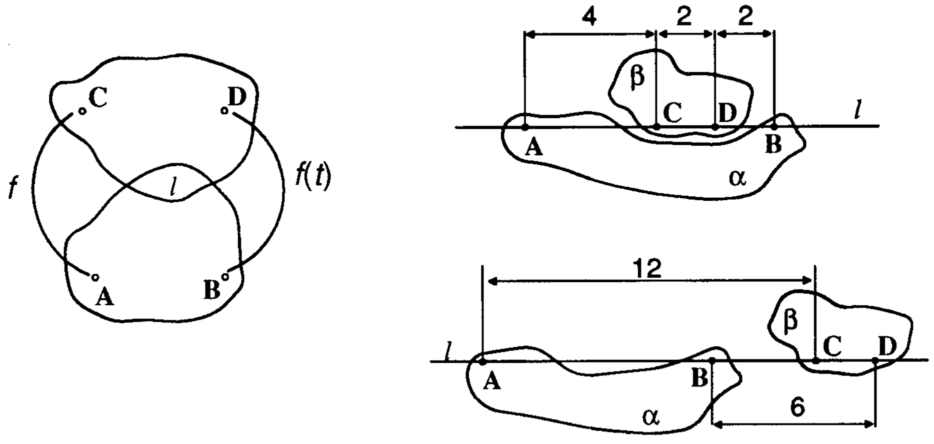

Figure 5: The dual solution of a simple *ed*-configuration

This situation must be accounted for by solvers which form a solution tree [1] and/or use heuristics to select the estimated user-desired configuration.

Two *ed*-clusters sharing a common circle or point

These two cases are practically equivalent as the degree of freedom of the second cluster wrt the first one is the angle of rotation around the common point or the centre of the common circle. We shall present the solution when a point is shared. Having a common circle may sometimes involve an additional scaling to the transformations applied to both clusters.

Let $\mathbf{S}$ be the only common element of clusters α and β having coordinates (S_x^α, S_y^α) and (S_y^β, S_y^β) respectively. As before, we first need to translate each cluster so that $\mathbf{S}$ coincides with the origin; fix the first cluster; and then apply a rotation to the second one with two unknowns $u = \cos(\theta)$ and $v = \sin(\theta)$ corresponding to the rotational degree of freedom θ.

This construction step is carried out by solving the following system of two multivariate quadratic polynomial equations:

$$\left| \begin{array}{l} \|\mathbf{A} - \mathbf{S}_\alpha - \mathbf{R}(u,v)\cdot(\mathbf{C} - \mathbf{S}_\beta)\| = \|\mathbf{B} - \mathbf{S}_\alpha - \mathbf{R}(u,v)\cdot(\mathbf{D} - \mathbf{S}_\beta)\| \\ u^2 + v^2 = 1 \end{array} \right. .$$

Computing the coefficients of the first polynomial we obtain the form:

$$\left(\|\mathbf{C} - \mathbf{S}_\beta\|^2 - \|\mathbf{D} - \mathbf{S}_\beta\|^2\right)\left(u^2 + v^2\right) + c_0 u + c_1 v + c_2 = 0$$

The product of the unknowns has a 0 coefficient and the system can be reduced to a much simpler one:

$$\left| \begin{array}{l} c_0 u + c_1 v + c_2 + c_3 = 0 \\ u^2 + v^2 = 1 \end{array} \right. ,$$

for which we can certify that four real solutions exist when

$$c_0^2 + c_1^2 - (c_2 + c_3)^2 > 0,$$

while a unique construction step can be taken only if either the above or c_0 equals 0. Otherwise, the *ed*-constraint in question is contradictory and the solver terminates.

The clusters are merged by transforming all elements of β by the matrix:

$$\mathbf{M}_{\beta\alpha} = \mathbf{T}(S_y^\alpha, S_y^\alpha)\cdot\mathbf{R}(u,v)\cdot\mathbf{T}(S_y^\beta, S_y^\beta).$$

Unlike the previous configuration, a more complex function than identity relating the pairs of points does not complicate the final system of equations, nor it increases the number of possible solutions.

Other configurations involving *ed*-constraints

As mentioned above, an equal distance might be specified between a line and a point and two parallel lines as well. The main difference from the distance between points of these two is that a line has orientation and the distance to it is usually signed. To avoid the treatment of various combinations, we have limited the construction patterns to involve only constraints of identical type.

This makes it impossible to require the distance between a point and a line to be equal to the distance between two points. However, for the needs of parametrising geometry made of clusters of similar elements and constraints this is not a serious limitation.

Other configurations, that can be of interest, occur if the two clusters do not share a common element, but are linked by three constraints, two of which are *ed*-constraints and the third is a *prescribed distance* or another *ed*-constraint. The most simplified case of this type is finding the point located at equal distances from three other points. A more general one is shown in Fig. 6.

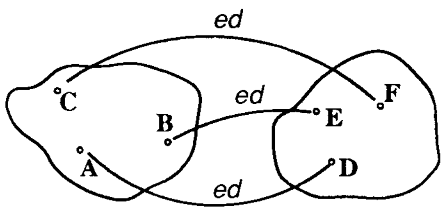

Figure 6: Two clusters linked by three *ed*-constraints

Unlike the *ed*-constraints the *angle constraint* may only involve two lines and the angle itself is always a signed value. Thus, the *ea*-constraints are signed as well.

There is only one basic pattern which is of particular interest: two *ea*-clusters sharing a common point. A single common line does not determine the exact position of the second cluster wrt to the first one as the former is still free to slide along the common line. We need to perform the same transformations as before and then use the algebraic equation representing a prescribed angle constraint. It is more convenient to equal the sine and the cosine rather than the angle itself.

Circularly constrained *ed*-clusters

In this section we present the solution of the more common case when several couples of geometric elements are linked by equal dimension constraints. As noted earlier, *ed*- and *ea*-constraints must be treated separately as they control different degrees of freedom. We concentrate on the equal distance constraint as it needs a more complicated approach depending on the common element, shared by the clusters.

Two cluster formations originate from the use of structural parameters: the first comprises a number of clusters all sharing a common element and each two adjacent being related by an *ed*-constraint; and a more general case when each two adjacent clusters possess a common element as illustrated in Fig. 7. The first configuration is the one produced as a result of modelling the COMB object as shown in Figures 2 and 3.

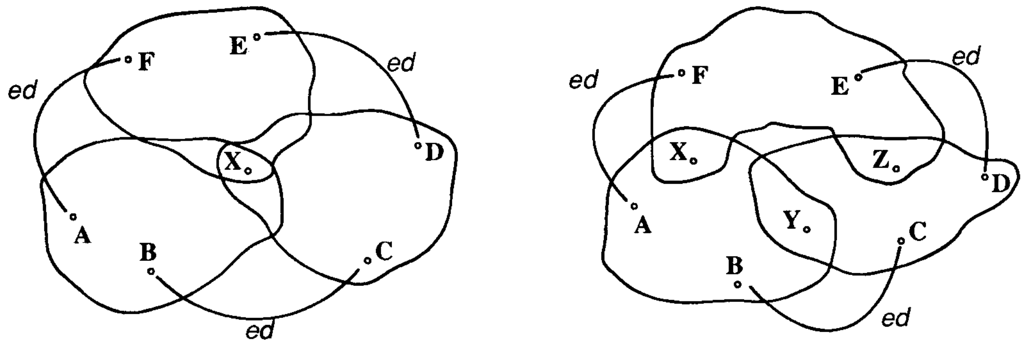

Figure 7: The two configurations of *ed*-cluster sets

In both cases we can apply the same location technique as presented above. Let us concentrate on the first, being the simpler situation. If the common element is a line the cluster relocation is translational, while if this element is a point the degree of freedom that the *ed*-constraint controls is the angle of rotation around the common point.

Let us first consider the presence of a common line. If we express the translational degrees of freedom u_i of each cluster β_i wrt to one of them α which we select as a basic one, we shall acquire the following system of equations:

$$\left| \; -t_i^2 + t_{i+1}^2 - a_i t_i + a_{i+1} t_{i+1} = c_i, \quad i = 1, \ldots, n \right.$$

wrt to the unknowns u_i given by $t_1 = u_1$, $\{t_i = u_i - u_{i-1},\ i = 2, \ldots, n\}, t_{n+1} = -u_1$.

This system can be solved symbolically once the coefficient are computed. For example, if the *ed*-constraints are imposed on pairs of points, a_i is the dot product expression:

$$a_i = 2\left(\mathbf{P}_i \cdot \mathbf{n}_i - \mathbf{Q}_{i-1} \cdot \mathbf{n}_{i-1}\right)$$

where the points $\mathbf{P}_i$ and $\mathbf{Q}_i$ belong to the cluster β_i, and $\mathbf{n}_i$ is the unit vector on the common line in the cluster local coordinate system. The *ed*-constraint relates the pairs $(\mathbf{Q}_{i-1}, \mathbf{P}_i), i = 0, \ldots, n$, where $\mathbf{P}_{n+1} = \mathbf{P}_0$ and $\mathbf{Q}_{n+1} = \mathbf{Q}_0$.

The symbolic computation produces all possible solutions, which are then analysed according to orientation of placement requirements or solver heuristics to select the one, expected by the user. A solution tree [3] may also be created as the transformation matrices are explicitly evaluated and stored. This permits the construction process to be revised and redirected by adding new selection criteria.

The same *transformation-based* approach is applied to the case of a common element being a point or a circle. The system to solve wrt to the rotational degrees of freedom of each cluster is more complex, but still solvable symbolically. A serious problem here is that it may have no real solution, i.e. a construction step might not be executable. Presently, we do not analyse the correctness of the imposed constraint but rather check for a real solution after the solver completes its task.

The situation is complicated substantially in the second type of configurations if the common elements are of different type, e.g. some clusters share a common point, while others share a common line. We have left this case aside as it is not produced by variable topology parametrisation. Due to the symmetry of constraints in parametric object the most common situation is the one shown at the left side in Fig. 7.

An example of nested structural parameters

The approach we have adopted is also suitable for processing graphs, where multiple sets of equal dimension constraints exists. There are two major cases when this occurs. The first is a configuration of two separate *ed*-cluster sets that can be solved independently. It might also happen that these two sets can not be separated, i.e. they form a single *ed*-cluster, between the members of which different *ed*-constraints exist. This requires the building of a more complex system of equations than the ones presented here. Again, this case is not typical for variable topology parametrisation.

The second case reflects the usage of nested structural parameters when representing a wider class of objects as the one shown in Fig. 8. The electrical machine depicted may have a parameter `Npoles` being the number of stator poles (8 for all instances shown) as well as a parameter `NslotsPerPole` to specify the number of slots on each stator pole.

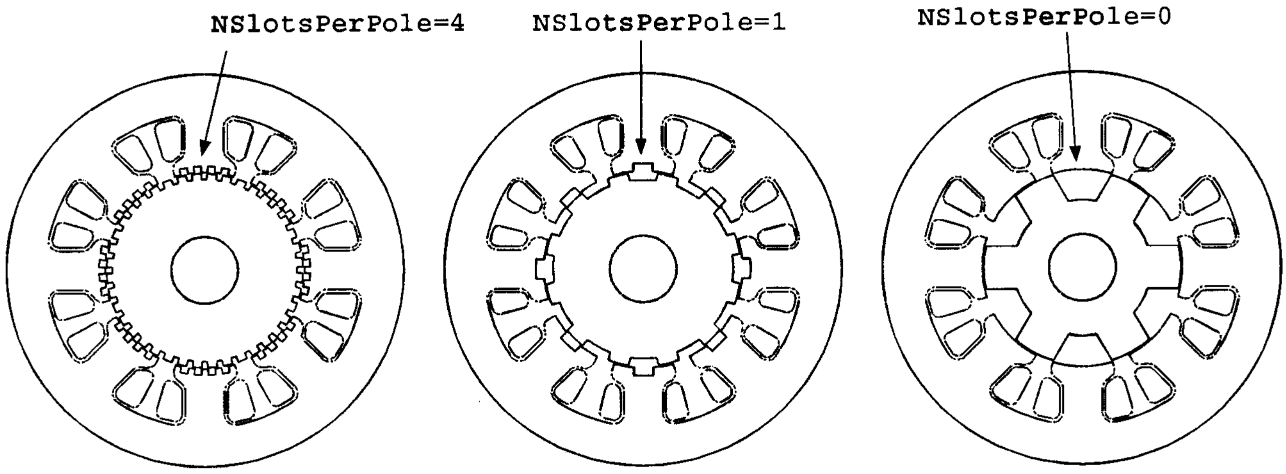

Figure 8: The effect of changing the value of a nested structural parameter

The bottom-up solving presented here will first detect the internal *ed*-clusters where the equal distance is imposed between the slots on each stator pole. The poles are then constructed independently of each other producing as a result a set of clusters, linked by the *ed*-constraints between the ends of the pole tips. The last step is then to solve this cluster pattern (where only one set of equal distance constraints is imposed) to produce the instance of the cross-section of the machine.

All cluster patterns in this example are of the type *sharing a common point or circle.* Together with the comb-like elements shown at the beginning, these represent the majority of variable topology parametric objects used. Another class of objects are the *matrix pattern features* (defined as such in the Step standard), which can be constructed in the same manner, because they can also be regarded as two nested *ed*-cluster sets.

To summarise, the system presented processes a parametric object in the following manner:

1. Generate the constraint graph for fixed values of all structural parameters. The result is a graph with fixed topology and will be altered only if a structural parameter's value is changed.
2. Replace all trivial equal dimension constraints by explicit ones. These are *ed-* (or *ea-*constraints) that link pairs, one of which has a prescribed distance (or angle) as well.
3. Repeat the following until no more free elements (clusters) exist.
   - 3.1 Create a new cluster by choosing an element with maximum links to other elements. Call it the *current cluster.*

3.2 For each element (cluster) not in the current cluster and linked to two elements in the latter by non-equal dimension constraints execute a construction step and join it to the current cluster.

3.3 If the current cluster has elements related to other graph nodes by equal dimension constraints check whether this cluster belongs to a solvable equal distance constraint pattern. If so, merge them in a single cluster.

Conclusion

We have presented the addition of *equal dimension* constraints that are necessary to model geometry with similar topology. The solvable patterns presented are restricted to the configurations that are inferred by constructing objects with repetitive structure. The future work is directed towards introducing preliminary top-down analysis to discover solvable equal dimension patterns aiming to verify correctness at an early stage and detect repetitive cluster configurations. A further challenge is to find a representation for the constraint graph avoiding the multiple solution of similar cluster configurations, which occur especially when nested structural parameters are used.

References

[1] B. Aldefeld. Variation of geometries based on a geometric-reasoning method. *Computer-Aided Design*, 30(3):65–72, 1988.

[2] A. H. Borning. The programming aspects of ThinkLab, a constraint-oriented simulation laboratory. *ACM TOPLAS*, 3(4): 353–387, 1981.

[3] W. Bouma, I. Fudos, C. Hoffmann, J. Chen and R. Page. A geometric constraint solver. Technical report CSD-TR-93-054, Purdue University, Computer Science, 1993.

[4] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. Technical report CSD-TR-93-076, Purdue University, Computer Science, 1993.

[5] D. C. Gossard and V. C. Lin. Representation of families of parts through variational geometry. In *Proceedings of IFIP Conference on Advances in CAD/CAM*, North-Holland, Amsterdam, pages 47–53, 1983.

[6] C. M. Hoffmann and R. Juan. Erep, a editable, high-level representation for geometric design and analysis. In P. Wilson, M. Wozny, and M. Pratt, editors, *Geometric and Product Modelling*, pages 129–164. North-Holland, 1993.

[7] E. N. Loukipoudis. Object management in parametric, programming-by-example system. In: *The Visual Computer*, 12(6):296–306, 1996.

[8] G. Nelson. Juno, a constraint-based graphic system. In *ACM SIGGRAPH, San Francisco, July 22-26*, pages 235–243, 1985.

[9] J. Owen. Algebraic solution for geometry from dimensional constraints. In *ACM Symp. Found. of Solid Modelling, Austin, TX*, pages 397-407, 1991.

[10] D. Roller. An approach to computer-aided parametric design. *Computer-Aided Design*, 23(5):385–391, 1991.

[11] L. Solano and P. Brunet. Constructive constraint-based model for parametric CAD systems. *Computer-Aided Design*, 26(8):614–621, 1994.

[12] I. E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference*, Spartan Books, Baltimore, MD, 1963.