

# A Constraint Hierarchies Approach to Geometric Constraints on Sketches

Christophe Jermann  
LINA - Université de Nantes, France  
christophe.jermann@univ-nantes.fr

Hiroshi Hosobe  
National Institute of Informatics, Japan  
hosobe@nii.ac.jp

## ABSTRACT

We propose an approach that uses preferences on the constraints in order to deal with over-constrained geometric constraint problems. This approach employs *constraint hierarchies*, a paradigm that has close relations with the traditional graph-based approaches used in geometric constraint solving. We also remark that any geometric constraint problem defined by imposing relations on a sketch becomes over-constrained as soon as the sketch is imposed as a *weak* constraint representing the designers intents. As a result our method appears very appropriate in CAD/CAM tools.

## 1. INTRODUCTION

Geometric modeling with constraints has been a hot topic of research for the past three decades and numerous methods have been proposed to handle the problems that arise in computer-aided design, robotics, and other application fields (see [3] for a survey). Most of the proposed methods consider that the problem is well-constrained (i.e., neither too many nor too few constraints), an assumption that does not hold very often in practice: the design is usually built incrementally and constraints are added and removed dynamically, yielding an evolving problem that is alternatively under-constrained and over-constrained until the design is finished. Recent developments [3] try to overcome these limitations. We propose a constraint hierarchies approach we think more convenient because:

- a user usually imposes constraints on a sketch that represents an initial guess of his intents and which can be naturally taken into account using our approach;
- preferences on constraints offer an additional flexibility in geometric modeling, avoiding the need to resort to complex debugging of non well-constrained problems;
- Geometric constraint solvers and constraint hierarchies solvers present several similarities that make their integration quite easy.

After briefly recalling the necessary background, we outline the proposed method and we illustrate its interest.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil  
Copyright 2008 ACM 978-1-59593-753-7/08/0003 ...\$5.00.

## 2. BACKGROUND

Geometric constraint problems involve geometric entities (e.g., points, lines, ...) on which geometric relations (e.g. distances, angles, ...) are imposed. Each entity has several *degrees of freedom (DOF)* and a constraint removes DOFs from the entities it constrains. Intuitively, entity DOF is the number of parameters required to fix its position, orientation and dimensions, while constraint DOF is the number of parameters it can fix. Hoffmann et al. [2] proposed a general graph-based approach (HLS) to geometric constraint solving. The problem is represented as a bipartite network with a source  $S$ , a sink  $T$ , one node per constraint  $c$  and entity  $o$ , arcs  $S \rightarrow c$  (resp.  $o \rightarrow T$ ) with capacities equal to the DOF of  $c$  (resp.  $o$ ) and arcs  $c \rightarrow o$  with infinite capacities linking constraints to the entities they constrain. A maximum flow in this network represents an optimal DOFs distribution, i.e., it determines which constraints fix which entities. HLS computes this flow incrementally in order to identify small solvable subproblems: the constraints are introduced one at a time in the network along with all relative arcs; the flow is updated and if it is perfect (all arcs  $S \rightarrow c$  and  $o \rightarrow T$  are saturated), an independent subproblem is identified; its constraints and entities are removed from the network, and the process is restarted.

*Constraint hierarchies* provide preferences (aka *strengths*) on constraints: the higher its strength, the more required the constraint. Constraints with the same strength belong to the same hierarchy level: they are equally preferred. Constraint hierarchies solutions depend on a criterion which defines how to handle the strengths. We focus on the *locally-predicate-better (LPB)* criterion which requires to satisfy as many constraints as possible. Hence LPB-solutions satisfy a maximal subset of constraints in each hierarchy level, which we call a *LPB-maximal* constraints subset. Gangnet and Rosenberg [1] proposed a general graph-based approach (GR) to constraint hierarchies considered with the LPB criterion. It identifies a LPB-maximal subset of constraints using an incremental maximum matching algorithm that matches first constraints with highest strength in a bipartite graph where the vertices are the constraints and the variables, and an edge links each constraint to each variable it constrains.

## 3. GEOMETRIC CONSTRAINT HIERARCHIES

We propose to adopt a constraint hierarchies approach to geometric constraint problems. This requires that each geometric constraint is associated with a strength, and that an adequate algorithm is designed to deal with these pref-

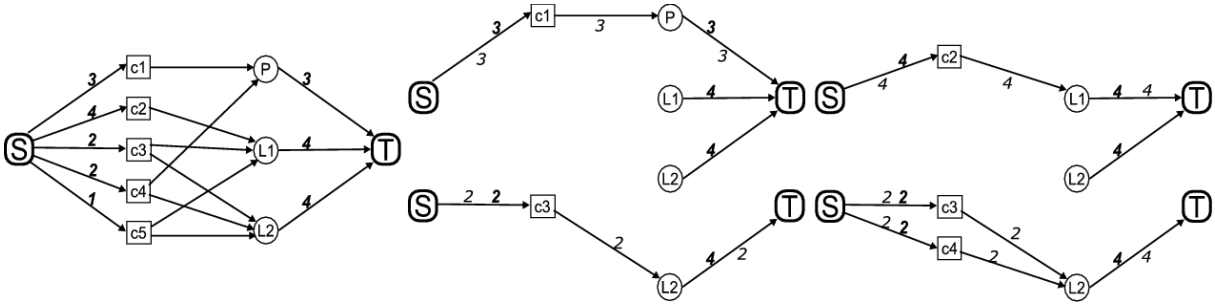


Figure 1: Application of our algorithm. Left: the bipartite network representing the problem. Right: four iterations of the algorithm.

erences.

Preferences are a powerful design lever in constraint-based design: the sketch on which a user usually imposes his constraints can be considered an initial guess of his intents and thus it can be used as weak *stay constraints* indicating it is preferred that the entities remain in the positions they have in the sketch; in addition, all the constraints imposed by the user can be considered preferential by default, letting the solver relax them as required in order to always return a solution in case the design is contradictory; finally, users could explicitly set strengths in order to achieve the desired solution instead of debugging contradictory specifications.

The method we propose consists in applying HLS introducing the constraints in decreasing strength order as done in GR. At each distribution, if the network is saturated, a cyclically dependent subset of constraints has been identified. Otherwise, if the lastly introduced constraint  $c$  could not be saturated, it cannot be satisfied and it is removed from the network; the flow is then restored to its previous state without  $c$  and the next constraint is considered. The algorithm stops when all the constraints have been tested. The constraints that were not removed then constitute a LPB-maximal subset; indeed, if a constraint could not be satisfied, this is due only to constraints with higher (or equal) strengths. This algorithm can be made incremental as GR in order to deal with interactive contexts. It is basically  $O(n^3)$  in time for a hierarchy with  $n$  constraints since computing a maximum flow requires quadratic time and this computation is done once for each constraint in the hierarchy at most. Hence, the algorithm .

**Example** In 3D, a point  $P$  (3 DOFs) and two lines  $L_1$  and  $L_2$  (4 DOFs each) are subject to five constraints:

name	strength	relation	DOFs
$c_1$	required	$fixed(P, 0, 0, 0)$	3
$c_2$	required	$fixed(L_1, 1, 0, 0, 0)$	4
$c_3$	strong	$parallelism(L_1, L_2)$	2
$c_4$	medium	$incidence(P, L_2)$	2
$c_5$	medium	$ortho\_distance(L_1, L_2, 4)$	1

Figure 1 represents the network corresponding to this problem and the execution of our algorithm:  $c_1$  is distributed first and the network is saturated, hence  $c_1$  can be solved separately to fix  $P$ ; similarly, the distribution of  $c_2$  indicates it can be solved separately to fix  $L_1$ ;  $c_3$  is distributed but not saturated, so  $c_4$  is distributed too, saturating the network:  $c_3$  and  $c_4$  together will fix  $L_2$ .  $c_5$  is not distributed since all objects have already been removed: it cannot be saturated. The LPB-maximal subset of constraints is then

$\{c_1, c_2, c_3, c_4\}$ . Note that the order in which the constraints in a hierarchy level are distributed can influence the result: if  $c_5$  was distributed before  $c_4$ , then  $c_4$  could not have been saturated and would be neglected, but yet the last subproblem  $\{c_3, c_5\}$  could not fix  $L_2$  (only 3 DOFs fixed by  $c_3$  and  $c_5$  against 4 DOFs for  $L_2$ ). Hence, the induced LPB-maximal subset of constraints  $\{c_1, c_2, c_3, c_5\}$  would be under-constrained (infinitely many solutions). Such cases cannot be dealt with using solely the LPB criterion.

## 4. CONCLUSION

The method we propose provides designers with an additional expressivity when defining interactively a geometric model: the sketch can be taken into account naturally and preferences can be tuned in order to achieve the right model without stumbling upon solver failures or complex specification debugging.

This paper provides only an overview of the method and the development potential is important: first the details and subtleties of geometric constraint solving methods have been neglected for the sake of clarity but they must be reintegrated to fully define the method; second, our method can in principle be applied to any constraint hierarchy and is not limited to geometric applications. It will be integrated into a general constraint hierarchies library developed by one of the authors, which will allow us to test it against problems from diverse application fields. An extensive comparison to existing will also be performed. At first sight, our algorithm seems to subsume both GR and HLS which are both recognized methods in their respective fields. The properties of our algorithm also have to be established and formalized. Its graph-based nature will certainly allow for a very precise study.

## 5. REFERENCES

- [1] M. Gangnet and B. Rosenberg. Constraint programming and graph algorithms. *Annals of Mathematics and Artificial Intelligence*, 8(3-4):271-284, 1993.
- [2] C. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition plans for geometric constraint systems. In *Proc. J. Symbolic Computation 2000*, 2000.
- [3] C. Jermann, G. Trombettoni, B. Neveu, and P. Mathis. Decomposition of geometric constraint systems: a survey. *International Journal of Computational Geometry and Applications*, 16(5-6):379-414, 2006.