Hierarchical Nonlinear Constraint Satisfaction

Hiroshi Hosobe
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
hosobe@nii.ac.jp

ABSTRACT

Constraint programming is a method of problem solving that allows declarative specification of relations among objects. It is important to allow preferences of constraints since it is often difficult for programmers to specify all constraints without conflicts. In this paper, we propose a numerical method for solving nonlinear constraints with hierarchical preferences (i.e., constraint hierarchies) in a least-squares manner. This method finds sufficiently precise local optimal solutions by appropriately processing hierarchical preferences of constraints. To evaluate the effectiveness of our method, we present experimental results obtained with a prototype constraint solver.

Keywords

constraint satisfaction, constraint hierarchies, nonlinear constraints, least squares

1. INTRODUCTION

Constraint programming, a method of problem solving that allows declarative specification of relations among objects, has been studied in various fields such as artificial intelligence, logic programming, and user interfaces. A use of constraint programming is geometric object layout in user interface applications such as drawing editors, since it is natural to use constraints to represent geometric relations among graphical objects.

It is important to allow preferences of constraints in constraint programming especially for user interface applications. This is because it is difficult for programmers to specify all constraints without conflicts. Typically, they need to provide graphical objects not only with constraints on their geometric layout, but also with other various constraints, for example, on their possible positional ranges, on their default locations, and on their movements by mouse dragging.

To handle preferences of constraints, the theoretical framework known as *constraint hierarchies* [5] has been widely used. In a constraint hierarchy, each constraint is assigned a multi-level hierarchical preference called a *strength*, which is often symbolically expressed as `required`, `strong`, `medium`, or `weak`, and solutions are determined in such a way that they satisfy as many strong constraints as possible. For example, the hierarchy of the constraints `strong` $x = 0$ and `weak` $x = 100$ yields the solution $x = 0$.

Recent progress of methods for constraint hierarchies has enabled use of *nonlinear constraints* such as Euclidean geometric constraints, nonoverlap constraints, and graph layout constraints [13, 14]. The power of such nonlinear geometric constraints is expected to further enhance the possibility of constraint-based user interface applications.

To our knowledge, however, the current nonlinear solvers only approximately satisfy constraint hierarchies. For example, the nonlinear geometric constraint solver Chorus [13] finds layouts of objects that slightly differ from correct ones, usually by several pixels on computer screens. In fact, solving the hierarchy of `strong` $x = 0$ and `medium` $x = 100$, it obtains $x = 3.0303\cdots$ in a typical setting. The difficulty lies in the treatment of multi-level strengths in constraint hierarchies.

In this paper, we propose a new method for numerically solving hierarchies of nonlinear constraints in a least-squares manner. Our method is based on *weighted least squares* [10, 11], and computes sufficiently precise local optimal solutions by adopting a novel technique that we call the *hierarchical QR decomposition* and by combining it with the Gauss-Newton method. To evaluate the effectiveness of our method, we present experimental results obtained with a prototype constraint solver.

In this research, we restrict our attention to geometric object layout in user interface applications. However, our method is applicable to other kinds of applications that can be regarded as finding a solution to a hierarchy of nonlinear constraints.

The rest of this paper is organized as follows. Section 2 describes related work on constraint satisfaction. Section 3 explains weighted least squares on which our research is based. Section 4 proposes our method for handling hierarchies of nonlinear constraints, and Section 5 evaluates it by presenting experimental results. Section 6 discusses our method. Finally, Section 7 mentions the conclusions and future work of this research.

2. RELATED WORK

Most early methods for constraint hierarchies, such as the DeltaBlue constraint solver [8], handled local propa-

SAC'04, March 14-17, 2004, Nicosia, Cyprus

gation (or dataflow) constraints by using a graph-theoretic approach. Such methods have difficulty in processing simultaneous constraints, and are limited to simple problems.

Linear solvers such as Cassowary [1, 6], QOCA [6, 16], and HiRise [12] appropriately treat simultaneous situations in constraint hierarchies by linear or quadratic programming. However, they are not applicable to nonlinear constraints.

To solve hierarchies of nonlinear constraints, approximate methods have recently been developed. The Chorus constraint solver [13] approximately processes constraint hierarchies by combining nonlinear optimization with a genetic algorithm. In [14], a method is presented that handles complex graphical constraints such as nonoverlap constraints by dynamically approximating them by linear constraints. To our knowledge, however, there are no methods proposed that solve hierarchies of nonlinear constraints as precisely as our method presented in this paper.

Geometric constraint solvers have been studied in the fields of geometric reasoning and computer-aided design [15, 17]. A major approach in these fields is to satisfy constraints in a "constructive" fashion by appropriately placing geometric objects step by step according to the analysis of constraint problems. Although it achieves fast and accurate constraint satisfaction, it is limited in handling preferences of constraints.

Preferences of constraints can be treated by other frameworks than constraint hierarchies. For example, partial constraint satisfaction [9] and semiring-based constraint satisfaction [3] are sufficiently general to handle constraint hierarchies. Also, in [2], an approach to handling nonlinear constraints in such a general framework is presented. However, we restrict our attention to constraint hierarchies in this research since they are widely used and practically important in the field of user interfaces.

3. WEIGHTED LEAST SQUARES

This section describes the weighted least squares [10, 11] on which our method is based.

3.1 The Gauss-Newton Method

Gulliksson et al. proposed a hybrid algorithm [10] that combines the *Gauss-Newton method* and a generalized Newton-Raphson method to solve least squares problems concerning weighted sums of squares of nonlinear functions. In this paper, we direct our attention to the Gauss-Newton.

The method handles a least squares problem that minimizes an objective function defined with a vector function

$$\boldsymbol{f} = (f_1, f_2, \ldots, f_m)^T$$

consisting of m functions $f_i : R^n \to R$, and a diagonal matrix

$$W = \mathrm{diag}(w_1, w_2, \ldots, w_m)$$

with m nonnegative reals w_i such that $w_i \geq w_j$ for $i < j$, as follows:

$$\min_{\boldsymbol{x}} \frac{1}{2} \left\| W^{1/2} \boldsymbol{f}(\boldsymbol{x}) \right\|^2, \tag{1}$$

where

$$\left\| W^{1/2} \boldsymbol{f}(\boldsymbol{x}) \right\|^2 = \sum_{i=1}^{m} w_i f_i^2(\boldsymbol{x}),$$

which is multiplied by $1/2$ here by convention.

The problem (1) could be regarded as an ordinary least squares problem if it included weights w_i into functions $g_i(\boldsymbol{x})$ by letting $g_i(\boldsymbol{x}) = \sqrt{w_i} f_i(\boldsymbol{x})$. However, it is expected that (1) will allow a better numerical stability of a resulting algorithm when there exists a wide variety of weights. In addition, required constraints expressed with $f_i(\boldsymbol{x}) = 0$ will be realized by allowing weights w_i with infinite values, which means that two-level constraint hierarchies can be handled.

To solve (1) with the Gauss-Newton method, its k-th iteration solves the following linear least squares problem defined with the temporary solution $\boldsymbol{x}_k$:

$$\min_{\boldsymbol{p}_k} \frac{1}{2} \left\| W^{1/2} \{ J(\boldsymbol{x}_k) \boldsymbol{p}_k + \boldsymbol{f}(\boldsymbol{x}_k) \} \right\|^2, \tag{2}$$

where J is the $m \times n$ Jacobian matrix of $\boldsymbol{f}$. This problem yields the search direction $\boldsymbol{p}_k$, and then, by calculating the steplength α_k, it obtains the temporary solution

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$$

for the next iteration.

3.2 Modified QR Decomposition

Gulliksson et al. solve the linear least squares problem (2) as follows. First, the following *system equation*, which is equivalent to (2), is introduced:

$$\begin{bmatrix} M & J \\ J^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{p} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f} \\ 0 \end{bmatrix}, \tag{3}$$

where $M = W^{-1}$, and the arguments $\boldsymbol{x}_k$ of $\boldsymbol{f}$ and J and the index k of $\boldsymbol{p}_k$ are omitted for simplicity.

To solve the system equation (3), the *modified QR decomposition* [11] of J is adopted. Using an $m \times m$ matrix Q that is M-invariant (i.e., $QMQ^T = M$), it decomposes J as

$$J\Pi = Q \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where Π is an $n \times n$ permutation matrix, and R is an $n \times n$ upper triangular matrix. This QR decomposition gives $\boldsymbol{p}$ satisfying (3), by computing

$$\boldsymbol{p} = -\Pi \begin{bmatrix} R^{-1} & 0 \end{bmatrix} Q^{-1} \boldsymbol{f}.$$

This QR decomposition is constructed from the following n matrix transformations: the i-th step transforms the $(m-i+1) \times (n-i+1)$ matrix A_i, which has been obtained from the previous step (for the first step, $A_1 = J$ is used), into a matrix whose first column has zeroes except for the top element. Specifically, given

$$M_i = \mathrm{diag}(w_i^{-1}, w_{i+1}^{-1}, \ldots, w_m^{-1}),$$

the i-th transformation is performed by using an M_i-invariant matrix Q_i such that $Q_i^2 = I_{m-i+1}$ (here and below we indicate by I_s the $s \times s$ identity matrix) and

$$Q_i A_i \Pi_i = \begin{bmatrix} \alpha_i & \boldsymbol{b}_i \\ 0 & A_{i+1} \end{bmatrix},$$

as well as an appropriate permutation matrix Π_i (which corresponds to column pivoting). The actual Q_i is defined as

$$Q_i = I_{m-i+1} - \frac{(\boldsymbol{a} + \alpha_i \boldsymbol{e}_1)(w_i^{-1} M_i^{-1} \boldsymbol{a} + \alpha_i \boldsymbol{e}_1)^T}{\alpha_i (\alpha_i + a_1)}, \tag{4}$$

where $\boldsymbol{a}$ is the first column of $A_i \Pi_i$, a_1 is the first element of $\boldsymbol{a}$, $\boldsymbol{e}_1$ is the column vector of size $(n-i+1)$ with 1 as its

first element and 0 as the others, and

$$\alpha_i = \mathrm{sign}(a_1)\sqrt{w_i^{-1}\boldsymbol{a}^T M_i^{-1}\boldsymbol{a}}.$$

The advantage of solving the system equation (3) by the modified QR decomposition is its numerical stability when there is a wide variety of the diagonal elements of M (i.e., the inverses of the weights). In that case, the matrix in (3) is ill-conditioned, which follows that standard methods for (3) will only find solutions with low precision. By contrast, the modified QR decomposition avoids such a numerical problem. In addition, it enables infinite weights by allowing zeroes as the corresponding diagonal elements of M, which results in a special form of Q_i.

4. OUR METHOD

This section proposes a method for handling hierarchies of nonlinear constraints, based on the weighted least squares.

4.1 Reformulation of Constraint Hierarchies

First, reformulating a constraint hierarchy as a least squares problem, we formalize what we handle by our new method. The method deals with a problem that consists of a finite number of levels containing equal-strength constraints in a similar way to an ordinary constraint hierarchy.

The least squares problem is formalized in the following way. Consider that each level is indexed by an integer k such that $0 \le k \le l$ for a given positive integer l (as in constraint hierarchies, we use level 0 for the strongest constraints, and levels with larger indices for weaker constraints, which is formally defined below). Also, for each level k, let m_k be the number of constraints at the level, and suppose that each j-th constraint at the level is represented as[1] $f_{k,j}(\boldsymbol{x}) = 0$. Then solutions $\boldsymbol{x}$ are defined as follows:

$$\min_{\boldsymbol{x}} \frac{1}{2}\sum_{k=0}^{l}\sum_{j=1}^{m_k}\sigma^{l-k} f_{k,j}^2(\boldsymbol{x}), \tag{5}$$

where σ is a positive real, which is a parameter used to express strengths.

The solutions to the least squares problem (5) approximate the ones to the similar constraint hierarchy that is solved with the criterion known as least-squares-better (LSB) [5, 6]. In particular, the limits of the solutions $\boldsymbol{x}$ as $\sigma \to \infty$ (which we use in our actual method) are equal to the LSB solutions of the hierarchy, unless there is inconsistency among constraints at level 0. Intuitively, this is because, for a sufficiently large σ, stronger constraints (i.e., with smaller k) are associated with larger weights σ^{l-k}, and therefore are more respected in the least squares.

This formulation of constraint hierarchies has close relation to that for the Chorus constraint solver [13]; both of them formalize constraint hierarchies by minimization of objective functions. The major difference is that the above formulation allows much more distinct strengths than that for Chorus. Since we actually use the case $\sigma \to \infty$, the above formulation realizes completely hierarchical strengths. By contrast, the formulation for Chorus considers only approximate strengths.

[1] An inequality constraint $f(\boldsymbol{x}) \ge 0$ can also be expressed in such an equality form, by adding a slack variable s and then letting $g(\boldsymbol{x}, s) = f(\boldsymbol{x}) - s^2$.

4.2 Hierarchical QR Decomposition

Our method for constraint hierarchies solves the least squares problem (5) in case $\sigma \to \infty$. More specifically, it realizes satisfaction of hierarchies of nonlinear constraints by replacing the modified QR decomposition in Gulliksson et al.'s Gauss-Newton method with a novel technique called the hierarchical QR decomposition and described below.

Assume that constraints are sorted in the strength order; that is, the objective function has the following form:

$$\boldsymbol{f} = (f_{0,1}, \ldots, f_{0,m_0}, f_{1,1}, \ldots, f_{1,m_1}, \ldots, f_{l,1}, \ldots, f_{l,m_l})^T.$$

Then we have

$$m = m_0 + m_1 + \cdots + m_l.$$

Also, by the correspondence of (5) to (1), we have the matrix W with weights and the matrix M with inverse weights in the following form:

$$W = \mathrm{diag}(\sigma^l I_{m_0}, \sigma^{l-1} I_{m_1}, \ldots, I_{m_l})$$
$$M = W^{-1} = \mathrm{diag}(\sigma^{-l} I_{m_0}, \sigma^{1-l} I_{m_1}, \ldots, I_{m_l}).$$

We need to consider $w_i^{-1}M_i^{-1}$ to determine Q_i. Choose k such that

$$\sum_{j=0}^{k-1} m_j < i \le \sum_{j=0}^{k} m_j,$$

and let

$$m'_k = \sum_{j=0}^{k} m_j - i + 1.$$

Then $w_i^{-1}M_i^{-1}$ is expressed as follows:

$$\begin{aligned} w_i^{-1}M_i^{-1} &= \sigma^{k-l}\ \mathrm{diag}(\sigma^{l-k} I_{m'_k}, \sigma^{l-k-1} I_{m_{k+1}}, \ldots, I_{m_l}) \\ &= \mathrm{diag}(I_{m'_k}, \sigma^{-1} I_{m_{k+1}}, \ldots, \sigma^{k-l} I_{m_l}). \end{aligned}$$

Although Q_i could be determined by using this $w_i^{-1}M_i^{-1}$ in (4), we do not use this form of Q_i here. In the hierarchical QR decomposition, we instead use Q_i^*, which corresponds to the limit of Q_i as $\sigma \to \infty$, that is,

$$Q_i^* = I_{m-i+1} - \frac{(\boldsymbol{a} + \alpha_i^* \boldsymbol{e}_1)\left(\begin{bmatrix} \boldsymbol{a}' \\ 0 \end{bmatrix} + \alpha_i^* \boldsymbol{e}_1\right)^T}{\alpha_i^*(\alpha_i^* + a_1)},$$

where $\boldsymbol{a}'$ is the column vector consisting of the first m'_k elements of $\boldsymbol{a}$, and

$$\alpha_i^* = \mathrm{sign}(a_1)\left\|\boldsymbol{a}'\right\|.$$

Note that, as $\sigma \to \infty$, the following holds:

$$w_i^{-1}M_i^{-1} \to \mathrm{diag}(I_{m'_k}, 0_{m_{k+1}+\cdots+m_l}),$$

where $0_{m_{k+1}+\cdots+m_l}$ is the $(m_{k+1}+\cdots+m_l)\times(m_{k+1}+\cdots+m_l)$ zero matrix.

Intuitively, the i-th transformation using Q_i^* distributes violations of linearly approximated constraints with strength k in a least squares manner, whereas it leaves weaker constraints for later transformations. These transformations are performed in the strength order since constraints are sorted beforehand.

An obvious advantage of using Q_i^* instead of Q_i is that the QR decomposition no more depends on the parameter σ.

Excluding σ is also expected to contribute to a better numerical stability of the resulting algorithm. Another benefit is that the QR decomposition is more efficient because of the simpler structure of Q_i^*.

5. EXPERIMENTAL EVALUATION

To evaluate the hierarchical nonlinear constraint satisfaction method proposed in the previous section, we implemented a prototype program in C++, and performed an experiment. In the experiment, we used a geometric layout problem that handles two constraint hierarchies in the following scenario.

Constraint hierarchy 1 (CH1): Suppose that a point (x, y) is located at $(150, 150)$. We newly constrain the point to be on the circle whose center is at the origin and whose radius is 100. This situation is expressed by the following constraint hierarchy:

$$\begin{array}{ll} \texttt{required} & \sqrt{x^2+y^2} = 100 \\ \texttt{weak} & x = 150 \\ \texttt{weak} & y = 150. \end{array}$$

Here we specify the circular positioning constraint as the required one, and also the information on the original position as the weak ones, by which we minimize the movement of the point. The theoretical solution of this hierarchy is $(x_1^*, y_1^*) = (100/\sqrt{2}, 100/\sqrt{2}) = (70.7\cdots, 70.7\cdots)$.

Constraint hierarchy 2 (CH2): Next, keeping the circular positioning constraint, an attempt is made to move the point (x, y) to $(-90, 120)$.[2] It is represented by the following constraint hierarchy:

$$\begin{array}{ll} \texttt{required} & \sqrt{x^2+y^2} = 100 \\ \texttt{strong} & x = -90 \\ \texttt{strong} & y = 120 \\ \texttt{weak} & x = x_1 \\ \texttt{weak} & y = y_1, \end{array}$$

where x_1 and y_1 indicate the solutions computed from CH1. Here we specify the movement constraint as the strong, which results in giving a stronger preference to the circular positioning constraint. The information on the original location of the point is unnecessary in this case; however, since it is generally unknown without constraint satisfaction, we leave the information as the weak constraints. The theoretical solution is $(x_2^*, y_2^*) = (-60, 80)$.

We compiled the program by using GCC 2.95.3 with the –O3 option, and executed it on a 1.13 GHz Pentium III-M processor running Linux 2.2.25. We used double precision for floating point arithmetic.

By solving these constraint hierarchies with this program, we obtained the results shown in Table 1. The errors indicated here are the distances $\|(x, y) - (x^*, y^*)\|$ between the computed solutions (x, y) and the theoretical ones (x^*, y^*). We also illustrate the changes of the temporary solutions during the iterations in Figure 1. These results indicate that our constraint satisfaction method processes these constraint hierarchies appropriately, obtaining sufficiently precise solutions for display on the computer screen.

[2]This situation is difficult for the constraint solver since the movement is more drastic than by ordinary mouse dragging.

Table 1: Results of the experiments on the satisfaction of the constraint hierarchies.

Hierarchies	Computational errors	Numbers of iterations	Computation times
CH1	2.6×10^{-14}	1	< 10 ms
CH2	2.9×10^{-3}	13	< 10 ms

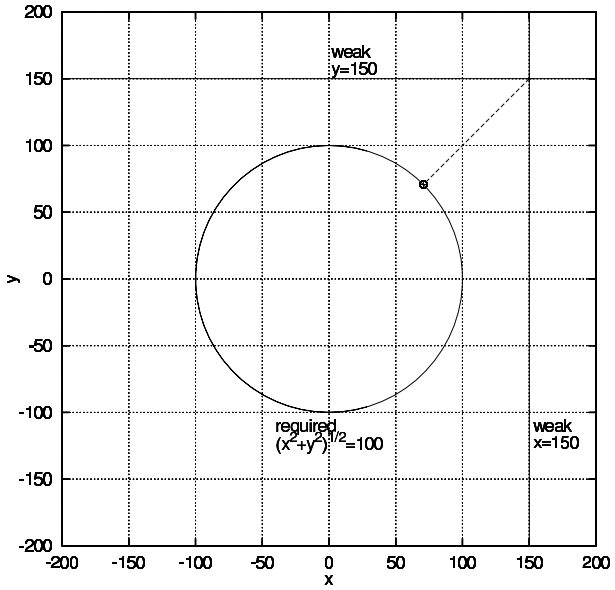

Figure 1: Changes of temporary solutions in the satisfaction of the constraint hierarchies (a) CH1 and (b) CH2.

6. DISCUSSION

Previous research on constraint hierarchies often focused on incremental constraint satisfaction [1, 6, 8, 12, 14, 16]. By contrast, we have not considered incrementality in this research. This is because nonlinear constraint satisfaction typically needs to considerably change elements of internal matrices, which weakens the effect of incrementality. However, if we can expect constraint hierarchies to be "almost linear" as in [14], incrementality will be useful for speeding up constraint satisfaction.

Our hierarchical nonlinear constraint satisfaction method is not guaranteed to obtain global optimal solutions. In general, numerically finding global optimal solutions is a hard problem. An approach to this problem is to combine a numerical method with a genetic algorithm [13], and is also applicable to our method.

Gulliksson et al. proposed combining the Gauss-Newton method with a generalized Newton-Raphson (GNR) method for nonlinear least squares, and showed that the GNR method reduces numbers of iterations when it is close to solutions [10]. The GNR method is expected to have a similar effect on hierarchical constraint satisfaction. However, we have not successfully extended the GNR method in such a way. Also, it should be noted that the GNR method requires computing Hessian matrices of constraint functions at each iteration, which is time-consuming in general. Therefore, it is unclear that the GNR method could reduce the total amount of computation time in ordinary applications.

A more promising direction is to develop a variant of quasi-Newton methods [4, 7] for hierarchical constraint satisfaction. Unlike the Newton-Raphson method (and the GNR method), a quasi-Newton method does not need Hessian matrices, and instead constructs approximate matrices by itself. It is known that quasi-Newton methods and their variants have been successful in many problems (e.g., sequential quadratic programming for constrained nonlinear optimization [7]). To develop such a method for hierarchical constraint satisfaction is a challenging open problem.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a method for solving nonlinear constraints with hierarchical preferences. The method finds precise local optimal solutions by combining the hierarchical QR decomposition with the Gauss-Newton method.

The next step of this research is to develop an easy-to-use constraint solver that adopts our method. We are also planning to implement several applications that use the solver to handle geometric constraints, to prove the effectiveness of our method by applying it to larger and more practical problems.

8. ACKNOWLEDGMENTS

We would like to thank the anonymous referees for their useful comments and suggestions. This research was supported in part by the Inamori Foundation Grant Program.

9. REFERENCES

[1] G. J. Badros, A. Borning, and P. J. Stuckey. The Cassowary linear arithmetic constraint solving algorithm. *ACM Trans. Comput.-Human Interact.*, 8(4):267–306, 2001.

[2] F. Benhamou and M. Ceberio. Soft constraints: A unifying framework applied to continuous soft constraints. In *Proc. Workshop of the ERCIM Working Group on Constraints and the CoLogNET Area on Constraint and Logic Programming*, 2003.

[3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *J. ACM*, 44(2):201–236, 1997.

[4] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[5] A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp Symbolic Comput.*, 5(3):223–270, 1992.

[6] A. Borning, K. Marriott, P. Stuckey, and Y. Xiao. Solving linear arithmetic constraints for user interface applications. In *Proc. ACM UIST*, pages 87–96, 1997.

[7] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2nd edition, 1987.

[8] B. N. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Comm. ACM*, 33(1):54–63, 1990.

[9] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artif. Intell.*, 58:21–70, 1992.

[10] M. Gulliksson, I. Söderkvist, and P.-Å. Wedin. Algorithms for constrained and weighted nonlinear least squares. *SIAM J. Optim.*, 7(1):208–224, 1997.

[11] M. Gulliksson and P.-Å. Wedin. Modifying the QR-decomposition to constrained and weighted linear least squares. *SIAM J. Matrix Anal. Appl.*, 13(4):1298–1313, 1992.

[12] H. Hosobe. A scalable linear constraint solver for user interface construction. In *Principles and Practice of Constraint Programming—CP2000*, volume 1894 of *LNCS*, pages 218–232. Springer, 2000.

[13] H. Hosobe. A modular geometric constraint solver for user interface applications. In *Proc. ACM UIST*, pages 91–100, 2001.

[14] N. Hurst, K. Marriott, and P. Moulder. Dynamic approximation of complex graphical constraints by linear constraints. In *Proc. ACM UIST*, pages 191–200, 2002.

[15] G. A. Kramer. A geometric constraint engine. *Artif. Intell.*, 58(1–3):327–360, 1992.

[16] K. Marriott and S. S. Chok. QOCA: A constraint solving toolkit for interactive graphical applications. *Constraints*, 7(3–4):229–254, 2002.

[17] J. C. Owen. Algebraic solution for geometry from dimensional constraints. In *Proc. ACM Solid Modeling*, pages 397–407, 1991.