Constraint Solvers for Graphical User Interface Applications

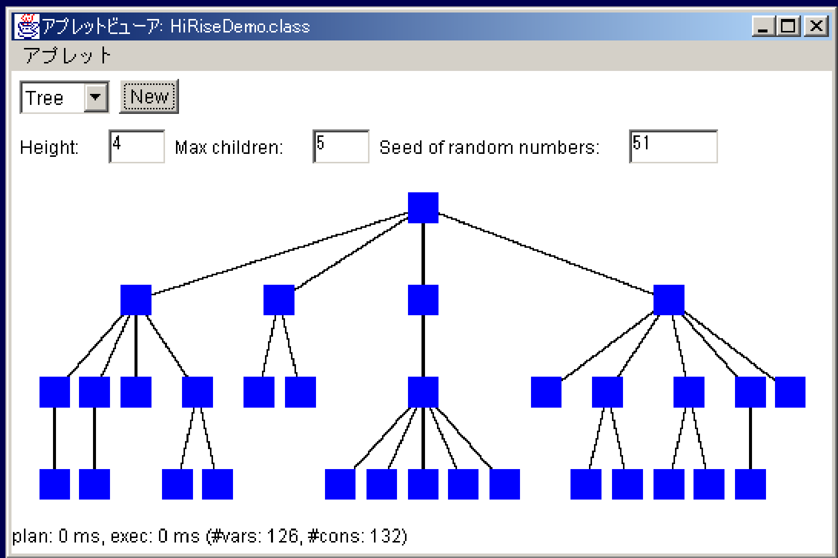
Hiroshi Hosobe

National Institute of Informatics

Background and Related Work

Constraints in Graphical User Interface (GUI) Applications

- Specify graphical layouts of objects.
- Automatically maintained by a constraint solver.



Constraints:

- The vertical distances between parents and children are equal
- The intervals of neighboring leaves are equal
- Subtrees sharing the same parents are adjacent
- Etc.

History of Constraint-Based GUI Applications

- SketchPad [Sutherland '63]
  - Pioneer
- ThingLab [Borning TOPLAS'81] [Borning & Duisberg TOG'86]
  - Provided modern graphical interfaces.
- ThingLab II [Maloney et al. OOPSLA'89]
  - Introduced constraint hierarchies.

Constraint Hierarchies [Borning et al. OOPSLA'87]

- A framework of soft constraints.
- Associate constraints with preferences called strengths.
  - Strengths are often symbolically expressed as required, strong, medium, or weak.
- Process inconsistencies among constraints.
  - Intuitively, satisfy as many strong constraints as possible.
- Ex. Hierarchy: strong x = 0, weak x = 1
  → Solution: x = 0

Constraint Hierarchies (contd.)

- Comparator: A criterion to process inconsistencies among equal-strength constraints
  - least-squares-better
    - Minimizes the sum of squares of constraint violations (least-squares method).
    - Ex. strong x = y, weak x = 0, weak y = 2

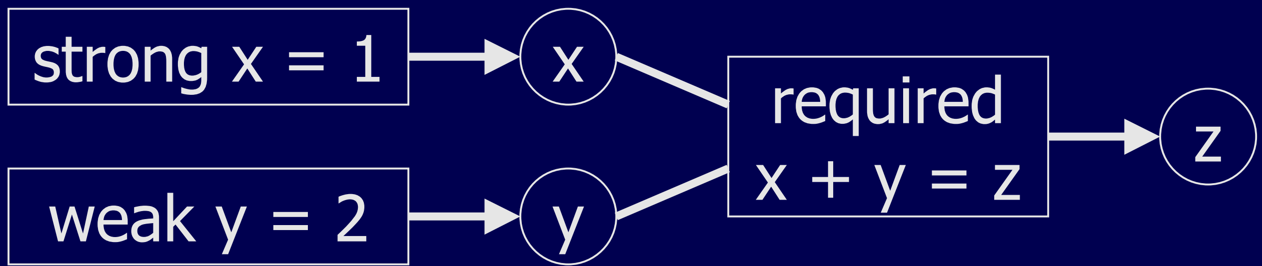
      → Solution: x = 1, y = 1
  - weighted-sum-better
    - Minimizes the sum of constraint violations.
  - locally-predicate-better/locally-error-better
    - Minimizes constraint violations in an arbitrary order.

Constraint Solvers for GUI Applications

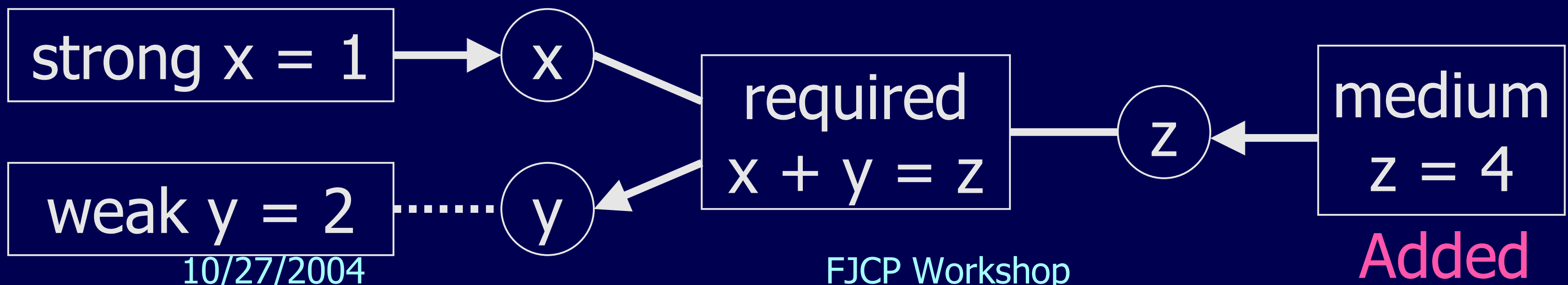
- Software that maintains and solves constraints.
  - Usually implemented as class libraries.
- Provide application programming interfaces:
  - Add new constraints.
  - Remove existing constraints.
  - Change variable values (typically to move graphical objects).

DeltaBlue [Freeman-Benson et al. CACM'90]

- An early constraint hierarchy solver.
- Incrementally solves hierarchies of local propagation constraints using locally-predicate-better.
  - Similar to the bipartite graph matching algorithm.
- Reports an error when it finds cyclic dependencies among constraints.
  - Cannot handle simultaneous constraints.



Sol.
x = 1,
y = 2,
z = 3



Sol.
x = 1,
y = 3,
z = 4

Cassowary [Borning et al. UIST'97]

- Incrementally solves hierarchies of linear equality/ inequality constraints using weighted-sum-better.
- Converts a constraint hierarchy into an optimization problem.
- Uses the simplex method.
- Still one of the most popular solvers.

required $x = y$
strong $y + 1 = z$
weak $x = 0$
weak $z = 3$

⇒

minimize $w_{strong}|e_1| + w_{weak}|e_2| + w_{weak}|e_3|$
subject to

$$x = y$$
$$y + 1 = z + e_1$$
$$x = 0 + e_2$$
$$z = 3 + e_3$$

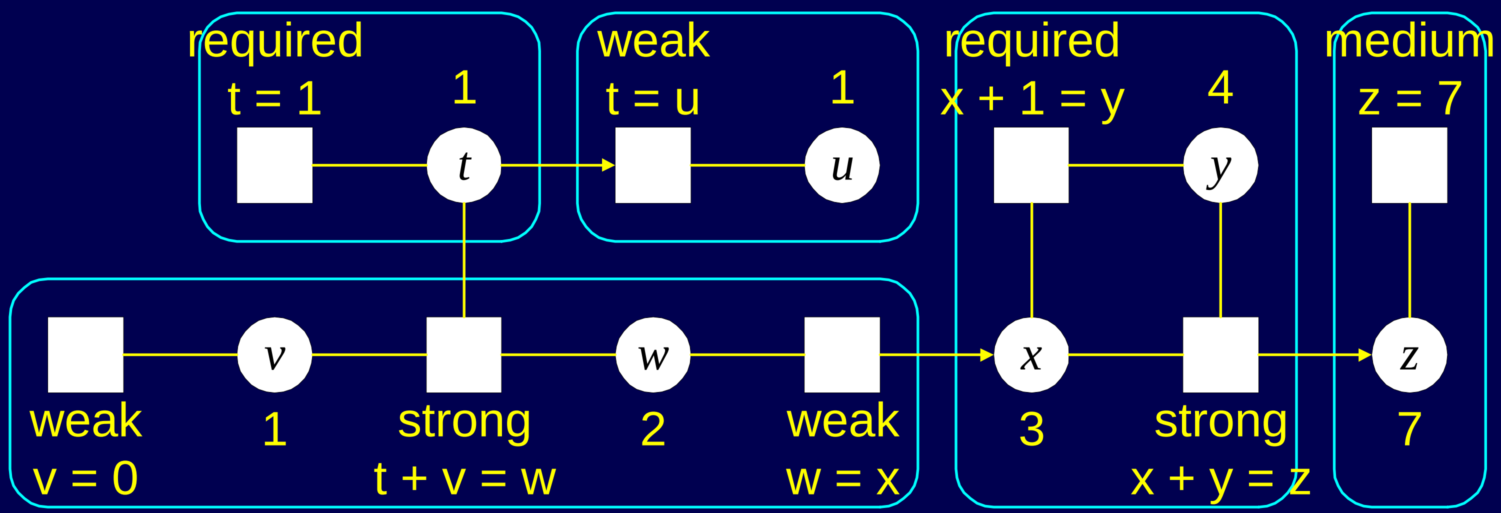
Our Research

DETAIL [PPCP'94, CP96]

- A graph-based algorithm that extends DeltaBlue
- "Generalized local propagation"
  - First local propagation algorithm that can handle, e.g., least-squares-better.

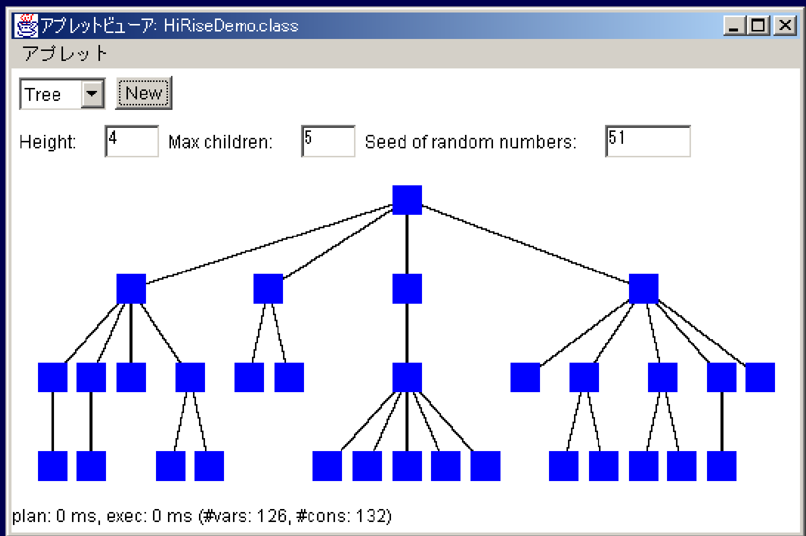
DETAIL (contd.)

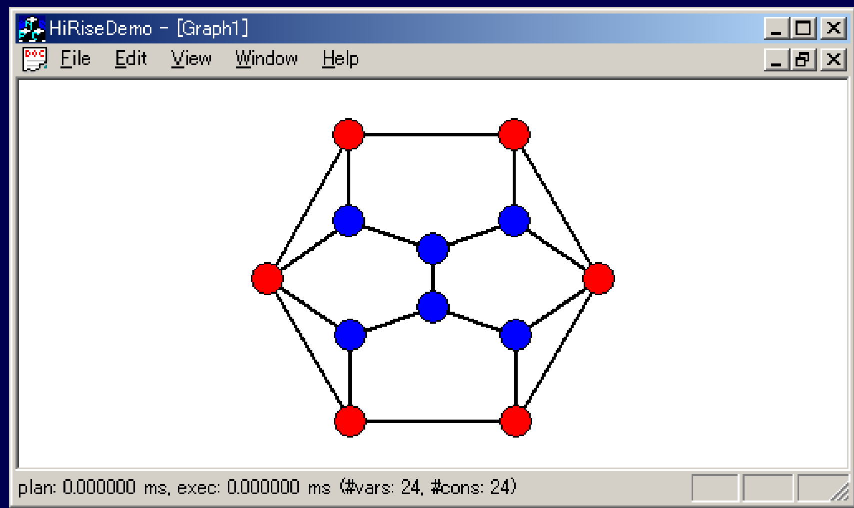
- Decomposes a constraint graph into "constraint cells."
- Generates a solution that respects the constraint hierarchy.



HiRise [CP2000]

- Maintains hierarchies of linear equality/inequality constraints.
- Solves thousands of constraints in real time.



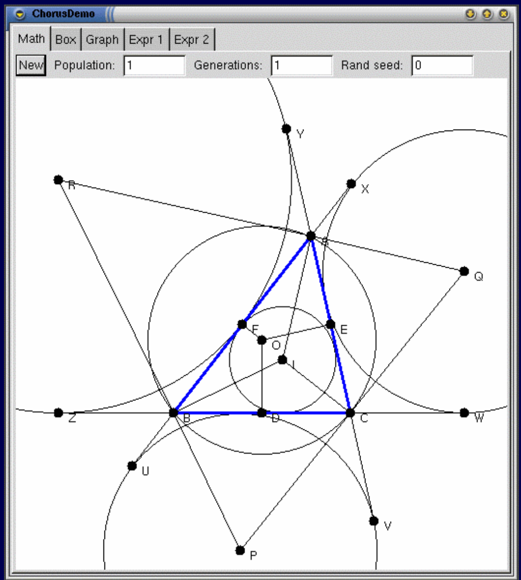


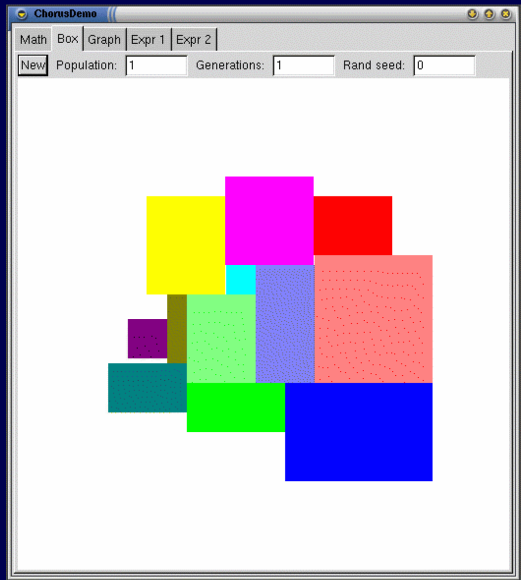
Algorithm of HiRise

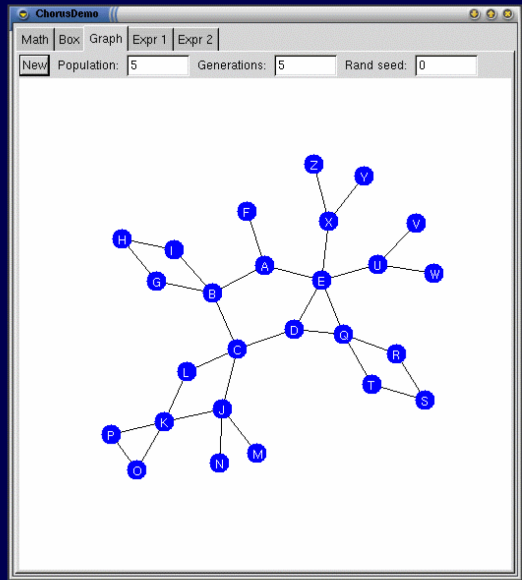
- Consists of two parts:
- Equality constraint processing
  - Classifies constraints into active and inactive ones.
  - Creates LU decomposition of active constraints.
  - Done incrementally.
- Inequality constraint processing
  - Adjusts the results of equality constraint processing to inequalities.
  - Based on the simplex method.

Chorus [UIST2001]

- Processes nonlinear geometric constraints:
  - Euclid geometric constraints (parallelism, perpendicularity, etc.)
  - Nonoverlap constraints
  - Graph layout constraints







Basic Framework of Chorus

- Represents constraints as error functions.
- Converts constraint hierarchies as optimization problems.
  - Defines an objective function as a sum of "weighted" constraint errors.
  - Represents constraint strengths as real-valued weights
- Approximately computes least-squares-better solutions.

Algorithm of Chorus

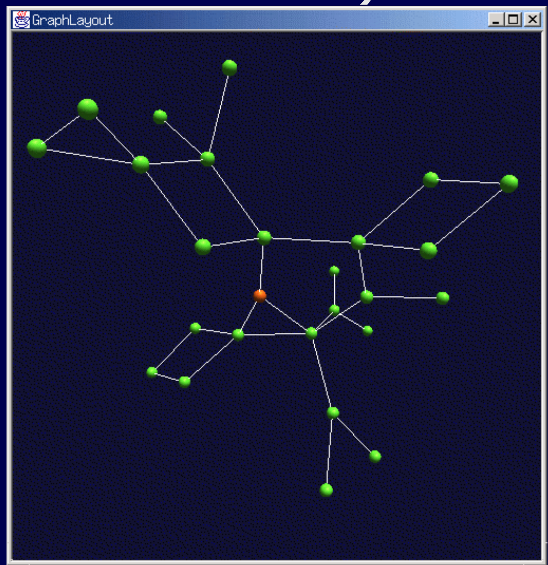
- Consists of four parts:
- Preprocessing required linear equality constraints to eliminate variables
- Local search by nonlinear numerical optimization
  - Typically using a quasi-Newton method
- Global search with a genetic algorithm
  - To obtain better solutions in a global sense
- Modifying constraint hierarchies
  - To cope with interactive operations and animation

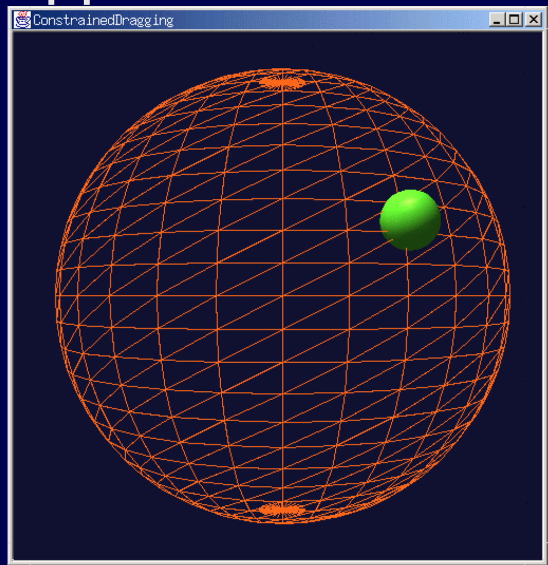
Module Mechanism of Chorus

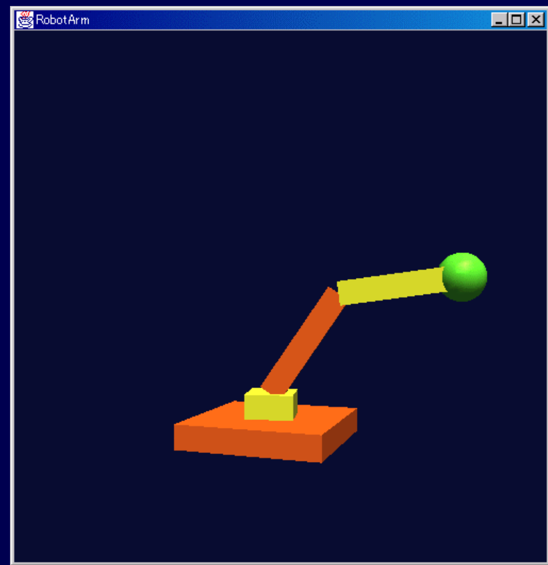
- Enables extension and modification of the solver.
- Evaluation modules
  - Calculates constraint errors.
  - Allows adding new kinds of constraints.
- Optimization modules
  - Performs numerical optimization.
  - Allows replacing optimization methods.

Chorus3D [Smart Graphics 2002]

- 3D version of the Chorus constraint solver.
- Supports hierarchies of coordinate systems in scene graphs.
- Applicable to geometric layout, constrained dragging, and inverse kinematics.
- Assumes Web3D technologies (e.g., Java 3D and VRML) as its applications







Processing Coordinate Transformations

- Model:
  - Each 3D point variable is associated with a local coordinate system.
  - Each coordinate transformation expresses its parameters as constrainable variables.
  - Each 3D geometric constraint
    - Refers to 3D point variables.
    - Defined as an "error function" and its gradient which uses world coordinates of 3D point variables.
- The solver provides a mechanism which embeds coordinate transformations in constraint error functions.
  - Transforms error functions and their gradients into the form using local coordinates and coordinate transformation parameters.

Sample Program: The Robot Arm Application

```
s = new C3Solver();
shldrTTfm = new C3TranslateTransform(new C3Domain3D(0, .1, 0));
s.add(shldrTTfm);
shldrRTfm = new C3RotateTransform(
        new C3Domain3D(0, 1, 0), new C3Domain(-10000, 10000));
s.add(shldrRTfm, shldrTTfm);
uarmTTfm = new C3TranslateTransform(new C3Domain3D(0, .1, 0));
s.add(uarmTTfm, shldrRTfm);
uarmRTfm = new C3RotateTransform(
        new C3Domain3D(0, 0, 1), new C3Domain(-1.57, 1.57));
s.add(uarmRTfm, uarmTTfm);
farmTTfm = new C3TranslateTransform(new C3Domain3D(0, .5, 0));
s.add(farmTTfm, uarmRTfm);
farmRTfm = new C3RotateTransform(
        new C3Domain3D(0, 0, 1), new C3Domain(-3.14, 0));
s.add(farmRTfm, farmTTfm);
handPos = new C3Variable3D(farmRTfm, new C3Domain3D(0, .5, 0));
editHandPos = new C3EditConstraint(handPos, C3.MEDIUM);
s.add(editHandPos);
editHandPos.set(getTargetWorldCoordinates());
s.solve();
double shldrAngle = shldrRTfm.rotationAngle().value();
double uarmAngle = uarmRTfm.rotationAngle().value();
double farmAngle = farmRTfm.rotationAngle().value();
```

construct a solver

specify a constraint system

suggest the target's position

solve the system

get the solutions

Limitation of Chorus

- Local optimal solutions are computed approximately.
  - To realize constraint strengths, it naively optimizes the sum of weighted squares of constraint violations.
  - It cannot use sufficiently distinct weights.
- Resulting object layouts are different from correct ones, usually by several pixels.

Hierarchy: strong $x = 0$, medium $x = 100$

Computed solution: $x = 3.0303\cdots$

A New Algorithm for Hierarchies of Nonlinear Constraints [SAC2004]

- Reformulates a constraint hierarchy as a weighted least-squares problem.
- Performs weighted nonlinear least squares by using a Gauss-Newton method
  - Using the weighted linear least squares based on hierarchical QR decomposition.

Reformulation of Constraint Hierarchies

- Uses weighted least-squares problems:

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

  - Variables: $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
  - Strengths: 0 (strongest), $1, \ldots, l$ (weakest)
  - Constraints: $f_{k,j}(\boldsymbol{x}) = 0$ (j-th constraint with strength k)
  - σ: positive real parameter to represent strengths

- The limits of solutions as $\sigma \to \infty$ are equal to least-squares-better solutions of the original hierarchy
  - Unless there is inconsistency among constraints with strength 0.

Algorithm

Weighted nonlinear least squares based on the Gauss-Newton method [Gulliksson et al. ’97]

- Repeatedly solves linear least squares:

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

  - $W = \mathrm{diag}(\sigma^l I m_0,\ \sigma^{l-1} I m_1,\ \ldots,\ I m_l)$
  - $\boldsymbol{f} = (f_{0,1},\ \ldots,\ f_{0,m_0},\ f_{1,1},\ \ldots,\ f_{1,m_1},\ ,\ldots,\ f_{l,1},\ \ldots,\ f_{l,m_l})$
  - J : Jacobian matrix of $\boldsymbol{f}$
- Proceeds to the next step by letting $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$.
  - α_k : steplength

Algorithm (contd.)

Linear least squares using hierarchical QR decomposition

- Based on modified QR decomposition [Gulliksson & Wedin '92].

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

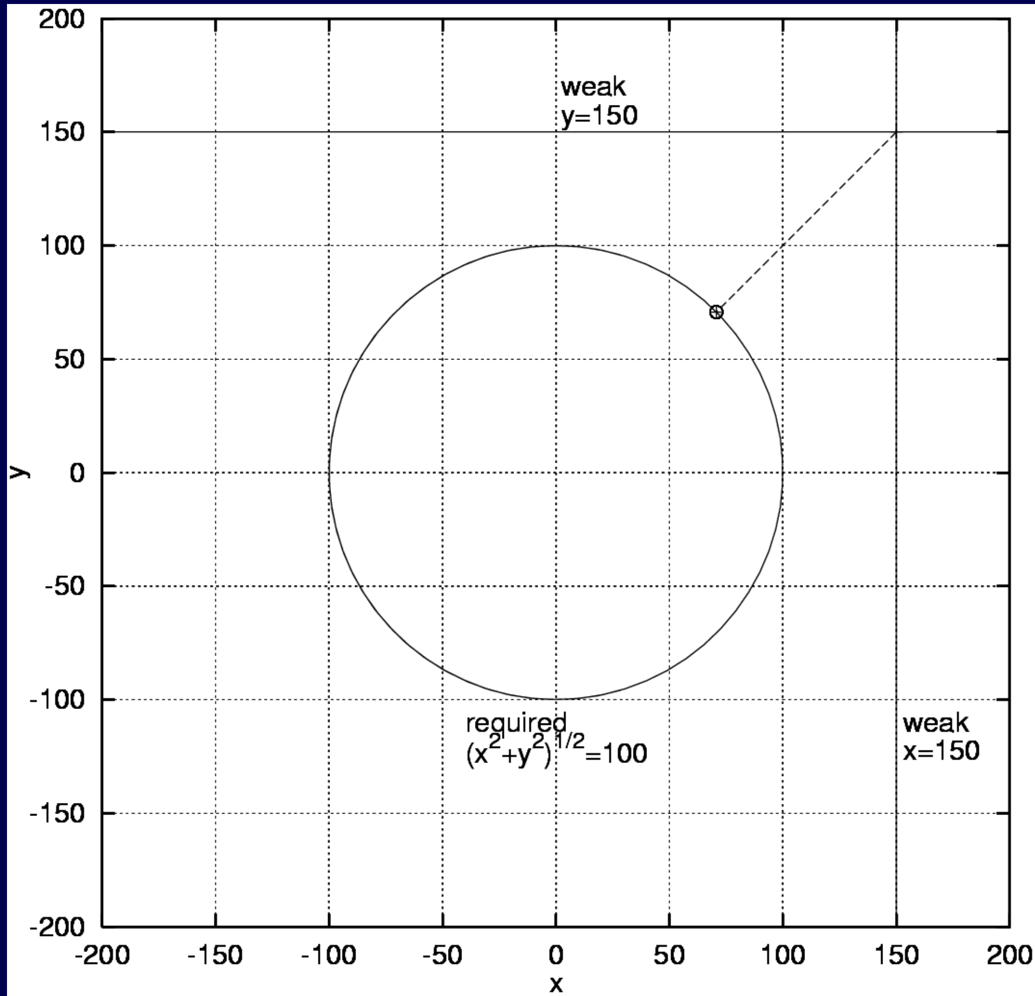
  - Π : permutation matrix (for column pivoting)
  - R : upper triangular matrix
- Obtains solutions corresponding to their limits as $\sigma \to \infty$.
- Processes J's rows corresponding to constraints one by one in the strength order.
  - Distributes violations of equal-strength constraints, leaving weaker constraints for later transformation.

Experiment: Solving Constraint Hierarchies (1)

- Constrain a point (x, y), which is initially at $(150, 150)$, to be on a circle.

required	$\sqrt{x^2 + y^2} = 100$
weak	$x = 150$
weak	$y = 150$

Error	2.6×10^{-14}
# of iterations	1
Time	< 10 ms



Experiment: Solving Constraint Hierarchies (2)

- Move (x, y) to $(-90, 120)$, keeping the circular positioning constraint.

required $\sqrt{x^2 + y^2} = 100$

strong $x = -90$

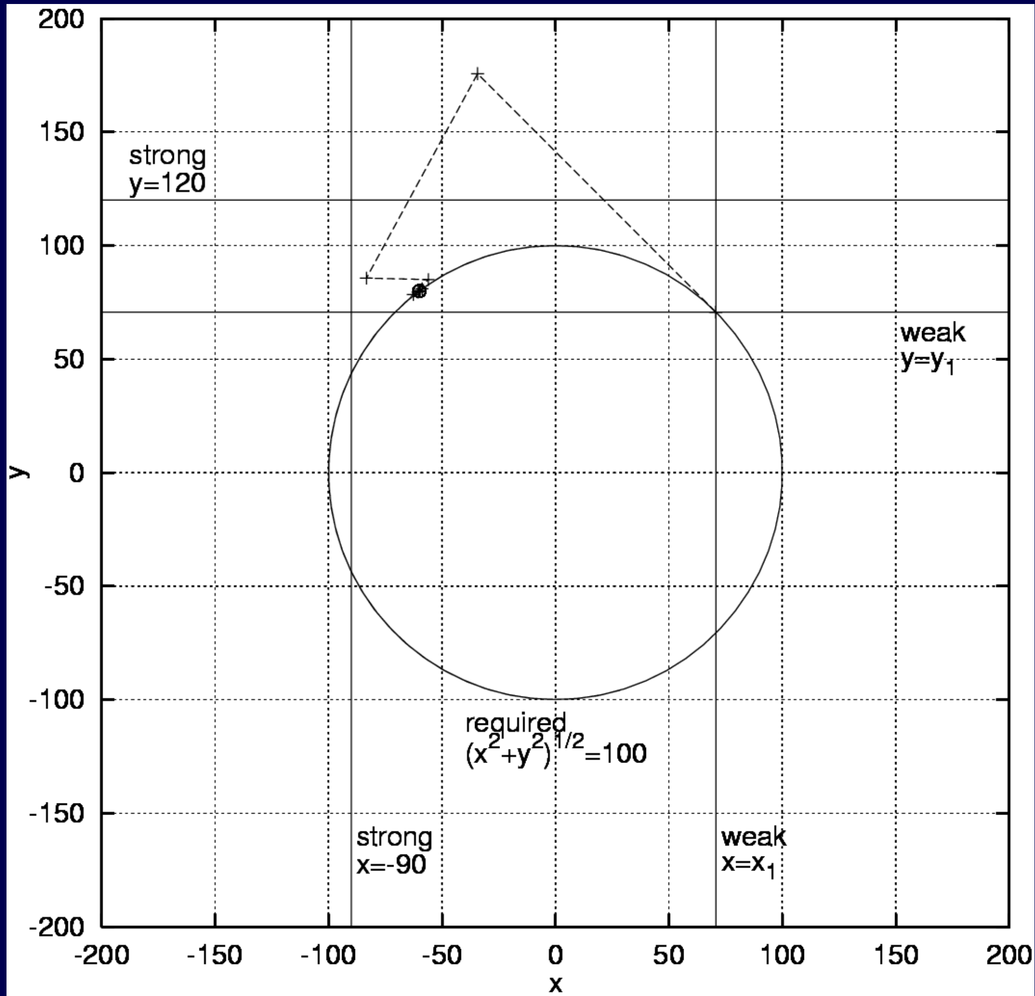
strong $y = 120$

weak $x = x_1$

weak $y = y_1$

(x_1 and y_1 indicate previously computed solutions)

Error	2.9×10^{-3}
# of iterations	13
Time	< 10 ms



Conclusions

- Constraints have been playing an important role in the graphical interface field since its infancy.
- Constraint hierarchies have been often used in graphical interface applications.
- Research on solving hierarchies of nonlinear constraints is still under way.
- Applying another soft constraint approach to graphical interface applications will be an interesting future direction.
  - E.g. continuous soft constraints.

Thank you very much