Soft and Interval Constraints for Layout of Diagrams

Hiroshi Hosobe
National Institute of Informatics, Japan
Outline of This Talk

- Soft constraints
  - Constraint hierarchies
  - Using constraint hierarchies for layout of diagrams
  - Methods for solving constraint hierarchies
- Interval constraints
  - Interval-based constraint hierarchies
Soft Constraints
Soft Constraints

- Constraints that can be “relaxed.”
- Useful for over-constrained problems.
  - Inconsistent constraints are satisfied as well as possible.
- Several approaches:
  - Least-squares method
    - Minimizes the sum of the squares of constraint violations.
  - Fuzzy constraint satisfaction
  - Semiring-based constraint satisfaction
  - 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.

- E.g. hierarchy: strong $x = 0$, weak $x = 1$
  $\rightarrow$ solution: $x = 0$
Constraint Hierarchies (contd.)

- Solved by a criterion called a comparator that processes inconsistencies among equal-strength constraints.

  - Least-squares-better
    - Uses the least-squares method.
    - E.g. strong $x = y$, weak $x = 0$, weak $y = 2$
      $\Rightarrow$ 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.
Using Constraint Hierarchies for Layout of Diagrams

- Required constraints are used to describe primary structures of objects.

- Strong constraints are typically used to describe less important objectives.
  - E.g. “pin” some object to some position.

- Medium constraints are typically used to describe desired motion of objects.
  - E.g. mouse dragging

- Weak constraints are typically used to prevent objects’ positions from sudden changes.

- Graph layout can be described with required or strong constraints.
Demo: The Chorus Constraint Solver [Hosobe ’01]

- Processes nonlinear geometric constraints:
  - Euclid geometric constraints (parallelism, perpendicularity, etc.)
  - Nonoverlap constraints
  - Graph layout constraints
Methods for Solving Constraint Hierarchies

- **Local propagation**
  - DeltaBlue [Freeman-Besnson et al. ’89]
  - SkyBlue [Sannella ’94]
  - DETAIL [Hosobe et al. ’94]
  - QuickPlan [Vander-Zanden ’96]

- **Linear methods**
  - Cassowary and QOCA [Borning, Marriott, Stuckey & Xiao ’97]
  - HiRise [Hosobe ’00]

- **Nonlinear methods**
  - Chorus [Hosobe ’01]
  - Chorus3D [Hosobe ’02]
  - [Hurst, Marriott & Moulder ’02]
  - [Hosobe ’04]
DeltaBlue
[Freeman-Benson et al. ’89]

- An early constraint hierarchy solver.
- Incrementally solves hierarchies of multi-way 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.

\[
\begin{align*}
\text{strong } x &= 1 \\
\text{weak } y &= 2
\end{align*}
\]

\[
\begin{align*}
x + y &= z \\
\text{required}
\end{align*}
\]

\[
\begin{align*}
x &= 1, \\
y &= 2, \\
z &= 3
\end{align*}
\]

\[
\begin{align*}
\text{strong } x &= 1 \\
\text{weak } y &= 2
\end{align*}
\]

\[
\begin{align*}
x + y &= z \\
\text{required}
\end{align*}
\]

\[
\begin{align*}
x &= 1, \\
y &= 3, \\
z &= 4
\end{align*}
\]

\[
\begin{align*}
z &= 4
\end{align*}
\]

\boxed{Added}
Cassowary
[Borning, Marriott, Stuckey & Xiao ’97]

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

\[
\begin{align*}
\text{minimize } & w_{\text{strong}} |e_1| + w_{\text{weak}} |e_2| + w_{\text{weak}} |e_3| \\
\text{subject to } & \\
\text{required } & x = y \\
\text{strong } & y + 1 = z \\
\text{weak } & x = 0 \\
\text{weak } & z = 3
\end{align*}
\]
Solving Hierarchies of Nonlinear Constraints

- Usually based on the optimization approach.

- Major issues:
  - Supported class of nonlinear constraints
  - Efficiency in constraint solving
  - Numerical accuracy of computed solutions
    - In both global and local senses.

- (To the speaker’s knowledge) all the sufficiently implemented solvers largely depend on some approximation of constraint hierarchies.

- Further research is necessary.
Chorus [Hosobe ’01]

- Handles nonlinear geometric constraints.

- Combines two methods:
  - Nonlinear numerical optimization for local search
    - Newton-based method
  - Genetic algorithm for global search
    - Obtains better solutions in a global sense.
    - Typically used only for initial solutions.

- Limitations:
  - Insufficient accuracy of local solutions
    - E.g. strong $x = 0$, medium $x = 100$
      $\rightarrow$ computed solution: $x = 3.0303\cdots$
  - No guarantee of global optimality
Interval Constraints
Interval Constraints

Combine two key technologies:

- **Interval analysis**
  - Variables are associated with intervals that include all their possible values.
  - Basic arithmetic operations are implemented with interval arithmetic.
    - \([l_1, u_1] + [l_2, u_2] = [l_1 + l_2, u_1 + u_2]\)
    - \([l_1, u_1] \times [l_2, u_2] = [\min(l_1 \times l_2, l_1 \times u_2, u_1 \times l_2, u_1 \times u_2), \max(l_1 \times l_2, l_1 \times u_2, u_1 \times l_2, u_1 \times u_2)]\)

- **Constraint propagation**
  - Constraints are used to eliminate impossible values.
    - E.g. constraint \(x + y = 5\) narrows variable value intervals 
      \(x = [0, 10], y = [0, 20]\) into \(x = [0, 5], y = [0, 5]\).
Branch-and-Reduce Algorithm

- Obtains a set of interval boxes that covers the solution set of a constraint problem.

- Main procedures:
  - Reduce: narrows a box by interval-based constraint propagation.
  - Branch: splits a box.

- Provides guarantees of solutions when terminated successfully:
  - Existence of a solution within a predetermined accuracy
  - Global optimality when applied to an optimization problem.
The Indigo Constraint Solver
[ Borning et al. ’96]

- Solves constraint hierarchies including inequalities.
- Extends the Blue algorithm [Maloney et al. ’89] to support inequality constraints.
  - By interval-based constraint propagation.
  - Cf. DeltaBlue is an incremental version of Blue.
- Not uses Branch-and-Reduce.
  - Simultaneous constraints could result in wide intervals (i.e. inaccurate solutions).
Interval-Based Soft Constraints [Benhamou & Ceberio ’03]

- An attempt to provide a unifying framework for soft constraints.
- Based on interval-based constraint optimization.
- Not powerful enough for handling constraint hierarchies.
Interval-Based Constraint Hierarchies

- Ongoing joint research project with Christophe Jermann, University of Nantes.

- Aims at finding “as-optimal-as-possible” solutions to constraint hierarchies in both global and local senses.

- Problem: The original formulation is not directly extensible to the interval-based definition of constraint hierarchy solutions.

- Our approach: An interval-based alternative formulation of constraint hierarchies.
  - Relaxes the condition for global optimality of solutions.
Conclusions

- Soft constraints
  - Constraint hierarchies
  - Using constraint hierarchies for layout of diagrams
  - Methods for solving constraint hierarchies
- Interval constraints
  - Interval-based constraint hierarchies

- Most of my papers are available at:
  - http://research.nii.ac.jp/~hosobe/pub-e.html