Computational Models of Constraint Propagation

Alexander V. Konstantinou
Columbia University

Candidacy Exam
December 9th, 1999

Presentation Overview

- Introduction
- Constraint Propagation (5)
- Constraint Logic Programming (3)
- Algorithms (6)
  - Interval Propagation (2)
- Systems (6)
- Future Work

Constraint Example

Variables: X, Y, U, Z
Domains: D_X = D_Y = D_U = D_Z = { white, red }
Constraints:
1. U = red
2. Y ≠ U
3. X ≠ Y
4. U ≠ Z

- Color flag (red, white)
- Maple leaf is red
- Neighbors have different colors

Formal CSP Definition

- Constraint is a relation over some domain D
- Constraint graph G = <C, V, D>
- Valuation function(v ˛ V) fi elements of D
- Solution S is set of all valuations satisfying all C

E.g., G= < { C*1.8 = f - 32 } , { c, f } >
S = { {c, f ! 0.0, 32.0 } … }

Constraint Graph Representation

F = (5/9)*C + 32

N-ary constraints

F = (5/9)*C + 32

Binary constraints

- How are constraints evaluated?
- F = (5/9)*C + 32 methods: (multi-way constraint)
  - F = (5/9)*C + 32
  - C = (5/9)*F + 32

Constraint Satisfaction

- Generate & Test (NP)
- Local Propagation (P)
  - No cycles (simultaneous equations)
  - No partial information constraints (greater-than)
- Search (NP)
  - E.g., Backpropagation + local propagation
- Domain-specific algorithms (P/ NP)
  - E.g., Gaussian elimination (integers)
Constraint Research

- Constraint Propagation
- Domain Specific Solvers
- Constraint Hierarchies
- Constraint Logic Programming
- CLP(Tree)
- CLP(FD)
- CLP(R)

Constraint Logic Programming

- CLP(Tree)
- CLP(FD)
- CLP(R)

December 9th, 1999
Alexander V. Konstantinou
7

Local Propagation

- Data-flow phases:
  - Determine variable value using constraint
  - Use value in another constraint, determine new variable value
- Handles non-numeric constraints
- Does not handle:
  - Cycles (simultaneous equations)
  - Partial information constraints (greater-than)

\[
\begin{align*}
X &= 3 \\
Y &= X+2 \\
Z &= X \cdot Y+1+X \cdot Y+2 \\
X &= 3 \\
Y &= 2 \\
Z &= Y+5
\end{align*}
\]

December 9th, 1999
Alexander V. Konstantinou
8

Backtracking + Propagation

[McAworth 1977, Mohr 1986]

- Node \( V_i \) is node consistent iff
  \[
  \forall x \in D_i, C_i(x)
  \]
- Arc(i,j) is arc consistent iff
  \[
  \forall x \in D_i, C_i(x) \exists y \in d_j, C_j(y) \land C_{ij}(x,y)
  \]
- Path \((i_0, i_1, \ldots, i_m)\) is path consistent iff
  \[
  \forall x \in D_{i_0}, y \in D_{i_m}, C_{i_0}(x) \land C_{i_m}(y) \land C_{i_0 i_1}(x, z_1) \land C_{i_1 i_2}(z_1, z_2) \land \ldots \land C_{i_{m-1} i_m}(z_{m-1}, y)
  \]

December 9th, 1999
Alexander V. Konstantinou
9

Spreadsheet Model [Zanden 92]

- Active-value-spreadsheet model
  - allow side-effects during constraint solving
  - solver decides ordering
  - cycle handling
- Procedures help solver
  - gain in efficiency
  - increase program complexity

December 9th, 1999
Alexander V. Konstantinou
10

Constraint Hierarchies

- Overconstrained/underconstrained problems
- Which variables to alter to satisfy multi-way constraints?
  - E.g., change IP host address, or renumber whole network?
- Constraint Hierarchies:
  - Labeled constraints (strength [0 … m])
  - Comparators (locally-better/globally-better)
  - Weights
  - Annotations (read/write only)

December 9th, 1999
Alexander V. Konstantinou
11

Constraint Logic Programming
Constraint Logic Programming

- The equality “1 + X = 3” fails in Prolog
  - Symbol ‘+’ considered unevaluated and unification fails
- Workarounds exist (use successor, or “is”)
- Solution: replace unification by constraints

solve([], C, C)
solve(Goal|Restgoal, Previous_C, New_C) :-
solve(Goal, Previous_C, Temp_C),
solve(Restgoal, Temp_C, New_C).
solve(Goal, Previous_C, New_C),
clause(Goal, Body, Current_C),
merge_constraints(Previous_C, Current_C, Temp_C),
solve(Body, Temp_C, New_C).

Expressive Power

First Order Predicate Calculus (FOPC)
  - Function Free FOPC
  - Horn FOPC
Propositional Calculus
  - Definite Clause Programs
  - Datalog
Finite Constraint Satisfaction
  - Constraint Satisfaction

Incremental Local Propagation

- Local propagation
- No cycles
- One-way constraints
- Incremental
- Handles constraint hierarchies
- Maintains solution graph
- Separates planning from evaluation
- O(M*N) | N constraints, M max methods/constr.
- Implemented in various systems

Multi-way Propagation (SkyBlue)

- General solver (methods)
- Incremental
- Cycle-aware
- Selects method, constructs directed method graph

Pointer Variables

- `node.value >= node.prev.value`
- Incremental algorithms (lazy/eager)
- Dependency graph based nullification/reevaluation
- Timestamps to support changing references during constraint evaluation
- One-way constraints
- Handles cycles
- O(|affected|)
### Inequality Constraints (Indigo)

- Acyclic graph
- Initially $[-\infty, +\infty]$ intervals
- Problem: division by zero
- Issue: single vs. multiple intervals
- Process strongest to weakest

\[
\begin{align*}
C &= A + B \\
10 &\leq A &\leq 20 \\
30 &\leq B &\leq 40
\end{align*}
\]

\[
\begin{align*}
A \text{.tighten}(C.\text{bounds} - B.\text{bounds}) &\quad A \in [10, 20] \\
B \text{.tighten}(C.\text{bounds} - A.\text{bounds}) &\quad B \in [30, 40] \\
C \text{.tighten}(A.\text{bounds} - B.\text{bounds}) &\quad C \in [40, 60]
\end{align*}
\]

- $O(n \times m)$ with $n$: variables, $m$: constraints
- One var. tightening per constraint (acyclic)

### Interval Propagation

- Label refinement (Waltz)
- Deductively sound
- Finite set $O(a \times e)$ with $a$: domain size, $e$: constraints
- Label languages, constraint languages

\[
\begin{align*}
\text{Con1: } X + Y = Z \\
\text{Con2: } Y \leq X
\end{align*}
\]

\[
\begin{align*}
Z \in [2, 7] \\
Y \in [3, 8] \\
X \in [1, 10]
\end{align*}
\]

### Systems & Constraint Impertative Programming

- Constraints + Object-Orientaiton

\[
\begin{align*}
&\text{solve type constraints} \\
&\text{execute constraint constructors} \\
&\text{solve primitive constraints}
\end{align*}
\]

### Constraints + Object-Orientaiton

- Goal:
  - Preserve flexibility of modern OO languages
  - Constraints on object methods
  - Maintain imperative OO style
  - Solve useful collections of constraints
  - Use refinement method (v.s. perturbation)

\[
\begin{align*}
\text{Point} &\quad \text{operator } \to \text{(Point p)} \\
\text{Point } x, y, z; &\quad x = y + z;
\end{align*}
\]

### Kaleidoscope’91

- New OO language (multiple dispatching)
- Types as constraints
- Time & Assignment
  - Pellucid values (current, previous)
  - Assignment: once constraint + weak stay
- Constraint constructors
  - Dispatch on each operator (no side effects)
Future Work

Constraints & Network Mgmt

- Object-relationship configuration model
- Under-constrained system
- Policy directed change propagation
- Domains: integers, strings, relations
- Constraints: equality, interval, set membership, ...
- Expressing constraints & propagation policies
  - Graphical language?