Constraint Hierarchy Networks

Roman Barták

Department of Theoretical Computer Science
Charles University, Prague

e-mail: bartak@kti.mff.cuni.cz
URL: http://kti.mff.cuni.cz/~bartak

Motivation

- why constraint hierarchies?
  - solving over-constrained problems
  - expressive power of preferential constraints
  - optimization techniques
- why hierarchy networks?
  - generalization of current constraint hierarchy solvers
  - framework for various algorithms
  - preserve planning & executing phases
  - solid theory behind

© Roman Barták, 1998

Constraint Hierarchies

- approach to solving over-constrained systems
- hard and soft constraints
  - hierarchy of labeled (preferential) constraints
- solution
  - stronger constrains are strictly preferred to weaker constraints
  - comparators (local, regional, global)
- relaxing of constraints (combining function)

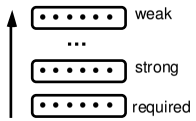
a=5@required, a+b=c@strong, c≤10@weak

© Roman Barták, 1998

Solving Constraint Hierarchies

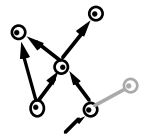
- refining method

  satisfy constraints at a hierarchy level together
  - \+ generality
  - \- non-incremental



- local propagation

  propagate values through a constraint graph
  - \+ fast, incremental
  - \- limited (functions, local comparator …)



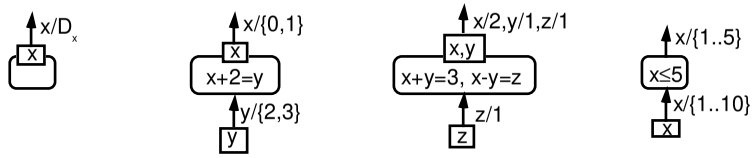
© Roman Barták, 1998

Constraint Cell

- finite non-empty set of equally preferred constraints with sets of input/output variables determined

potentially unsatisfied

free var | functional | generative | test

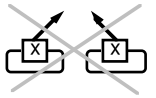


- encapsulation
  constraints in a cell are solved together without breaking the propagation principle

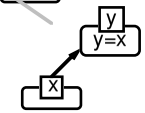
© Roman Barták, 1998

Constraint Hierarchy Network

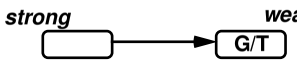
- directed acyclic graph where vertices are labelled by constraint cells and the following conditions hold:
  - conflict free decomposition to cells



  - edges from corresponding output to input



  - no weaker cell upstream potentially unsatisfied cell



  - no downstream forking in potentially unsatisfied cell

© Roman Barták, 1998

Solving Hierarchies Using Networks

- modular architecture
  - planning (construction of hierarchy network)
    - independent of comparator
    - dependent on constraint type (functions, relations)
  - executing (value propagation)
    - independent of hierarchy network
    - specific to constraints/domains and comparator
- one framework / various algorithms
  from refining method to local propagation
  Straton, Straton 2, BlueOne

© Roman Barták, 1998

Straton Algorithm

- pure stratification
- mimics refining method

- all equally preferred constraints (level) in one cell
- stronger cell determines the common variables
- "green" edges between cells with common variables
- "red" edges between subsequent levels

Algorithm (constraint addition):

1) put constraint to an equally preferred cell
   (if a new cell arises then add respective "red" edges)
2) change set of input/output variables and "green" edges accordingly

© Roman Barták, 1998

Straton 2 Algorithms - Features

- stratification with better dependencies
- improves (a bit) refining method

- all equally preferred constraints (level) in one cell
- stronger cell determines the common variables
- "green" edges between cells with common variables
- "orange" edges to remove downstream forkings

- algorithms for constraint addition and retraction
  - incremental part maintains "green" edges
  - common non-incremental part adds "orange" edges

© Roman Barták, 1998

Straton 2 Algorithms - Incremental Part

- maintains "green" edges only

the "no downstream forking" condition is not necessarily satisfied

Algorithm (constraint addition):

1) put constraint to an equally preferred cell
   (no "red" edges)
2) change set of input/output variables
   and "green" edges accordingly

Algorithm (constraint retraction):

1) remove constraint from respective cell
2) change set of input/output variables
   and remove invalidated "green" edges
3) find new cells determining removed variables
   and add necessary "green" edges

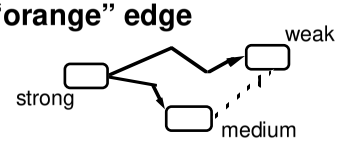
© Roman Barták, 1998

Straton 2 Algorithms - Non-Incremental Part

- add "orange" edges to remove downstream forkings

Algorithm:

- traverse cells (levels) in weak to strong manner
- compute cell's successors (both direct and indirect)
- if any downstream forking detected
  then add necessary "orange" edge



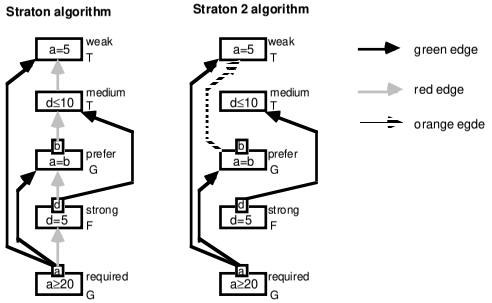
Worst complexity:

- depends on number of hierarchy levels only
- independent of number of constraints

© Roman Barták, 1998

Straton Family - comparison

- similarity
  vertices (cells) correspond to the hierarchy levels
- difference
  "red" vs. "orange" edges

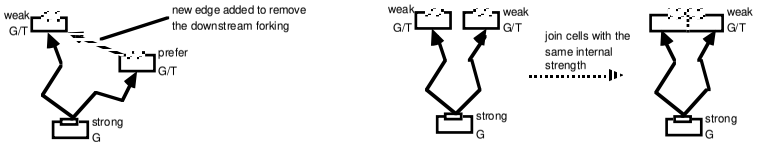


© Roman Barták, 1998

BlueOne Algorithm

- keeps constraint cells as small as possible
- based on ideas of DeltaBlue algorithm
  - one-output functional constraints
  - keeping global information on local place

    (walkabout preferences ...)
- two methods of removing downstream forkings



- no formal description yet

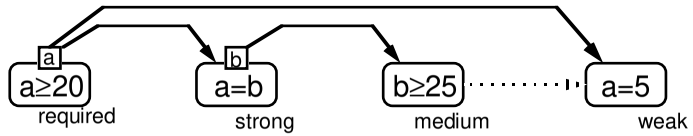
© Roman Barták, 1998

Value Propagation

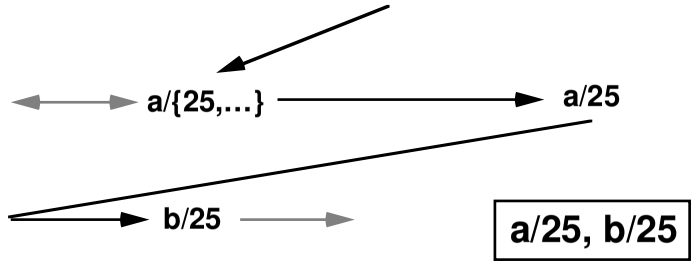
- *in theory:*
  - complete valuations are propagated
- *in practice:*
  - too many valuations
  - sets of variables' values are propagated
  - zig-zag method
- current implementation:
  - Indigo style of propagation via individual constraints
  - locally-better comparators
  - "green" edges for sound propagation of values
  - "red/orange" edges for sound ordering of cells

© Roman Barták, 1998

Value Propagation - Example



a/{20,...} → b/{20...} → b/{25,...}



© Roman Barták, 1998

Further Steps

- modification of DeltaOne algorithm for multi-output constraints
- value propagation
  - adaptation to finite domains
    consistency through one-way propagation
  - combination of solving techniques
    various domains, Gauss elimination, Simplex, ...
- application in a real-life project (scheduling)

© Roman Barták, 1998