Constraint Programming for Timetabling

Hana Rudová

Faculty of Informatics, Masaryk University
Brno, Czech Republic

`http://www.fi.muni.cz/~hanka`

ASAP Seminar, 2004

Purdue University Timetabling Problem

- Course demands for individual students
  - conflicts among classes of one student minimized
- Timetable for large lecture classes
  - Fall 2001
    - manually created: slightly smaller (750 classes), less constrained (about 40 assigned prior search)
  - Fall 2004
    - 830 classes $\doteq$ 1500 meetings
    - 50 classrooms
    - 89,633 course demands for 29,808 students
    - about 350 classes assigned before search
  - Spring 2005
    - easier data set
      no freshmen consideration, about 780 classes

Outline

Constraint Satisfaction Problem (CSP)

- Constraint satisfaction problem (X, D, C)
  - finite set of domain variables $X = \{V_1, \ldots, V_n\}$
  - finite set of values (domain) $D = D_1 \cup \ldots \cup D_n$
  - finite set of constraints $C = \{c_1, \ldots, c_m\}$
    - relations over subsets of variables

Constraint Satisfaction Problem (CSP)

- Constraint satisfaction problem (X, D, C)
  - finite set of domain variables $X = \{V_1, \ldots, V_n\}$
  - finite set of values (domain) $D = D_1 \cup \ldots \cup D_n$
  - finite set of constraints $C = \{c_1, \ldots, c_m\}$
    - relations over subsets of variables
- Partial assignment of variables $(d_1, \ldots, d_k), k \leq n$
- Complete assignment of variables $(d_1, \ldots, d_n)$
- Solution of CSP
  - complete assignment which satisfies all constraints

Constraint Satisfaction Problem (CSP)

- Constraint satisfaction problem (X, D, C)
  - finite set of domain variables $X = \{V_1, \ldots, V_n\}$
  - finite set of values (domain) $D = D_1 \cup \ldots \cup D_n$
  - finite set of constraints $C = \{c_1, \ldots, c_m\}$
    - relations over subsets of variables
- Partial assignment of variables $(d_1, \ldots, d_k), k \leq n$
- Complete assignment of variables $(d_1, \ldots, d_n)$
- Solution of CSP
  - complete assignment which satisfies all constraints
- Constrain & Search constraint propagation

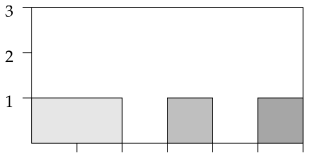
Constraint Satisfaction Problem (CSP)

- Constraint satisfaction problem (X, D, C)
  - finite set of domain variables $X = \{V_1, \ldots, V_n\}$
  - finite set of values (domain) $D = D_1 \cup \ldots \cup D_n$
  - finite set of constraints $C = \{c_1, \ldots, c_m\}$
    - relations over subsets of variables
- Partial assignment of variables $(d_1, \ldots, d_k), k \leq n$
- Complete assignment of variables $(d_1, \ldots, d_n)$
- Solution of CSP
  - complete assignment which satisfies all constraints
- Constrain & Search constraint propagation
- Timetabling example
  - variables: time T and classroom R for each class
  - domains: possible starting times, possible classrooms
  - constraints: required classrooms, precedence relations among times

Global Constraints

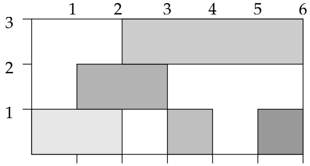
`serialized`

classes of the same instructor at different times



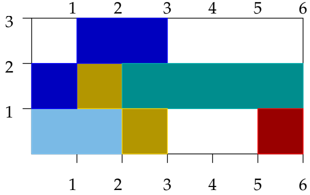
`disjoint2`

two classes must have different time or classrooms ⋆



`cumulative`

- for approximation (⋆)
  - – classes change classroom
  - – example: big classrooms
- on time variables only



Over-Constrained Problems

- Over-constrained problem
  - there is no solution with all constraints satisfied
- Solution:
  constraint modeling by hard and soft constraints
  - hard constraints = everything what must be satisfied
  - soft constraints = optimization part
    - part of the problem which can be unsatisfied
    - optimization for preferential requirements
- Example of soft constraints
  - too many preferential time requirements
  - two classes share some students

Soft Constraints

Weighted constraints

- each constraint associated with the weight
- example: $V_1 \# \backslash = V_2 @ \mathit{weight}$
- aim: minimize weighted sum of unsatisfied constraints

Soft Constraints

Weighted constraints

- each constraint associated with the weight
- example: $V_1 \# \backslash = V_2 @ weight$
- aim: minimize weighted sum of unsatisfied constraints

Weighted constraints + hard constraints

- stronger propagation for hard constraints
- optimization for weighted constraints

Solver for Soft Constraints

- Each value of the variable associated with a weight
  - unary soft constraints
- Soft constraint propagation
  - unsatisfied *c* @ *weight* increases weights of values for variables in *c*
- Evaluation of the solution $[V_1 = v_1, \dots, V_n = v_n]$:

$$\sum_{\forall i} w[V_i, v_i]$$

- Aim: minimize evaluation of the solution

Solver for Soft Constraints

- Each value of the variable associated with a weight
  - unary soft constraints
- Soft constraint propagation
  - unsatisfied *c* @ *weight* increases weights of values for variables in *c*
- Evaluation of the solution $[V_1 = v_1, \ldots, V_n = v_n]$:

$$\sum_{\forall i} w[V_i, v_i]$$

- Aim: minimize evaluation of the solution
- Example: $V_1 \# \backslash = V_2 @ weight$
  - V_1 or V_2 is instantiated to value v

  $\Rightarrow$ weight of the value v for second variable V_i increased by *weight* $\quad w[V_i, v] + weight$
  - partial forward checking

Examples of Soft Constraints

- Unary soft constraint
  - times or locations are preferred or discouraged
- Soft `serialized` constraint
  - class A should not overlap with B
  - weight is the number of students in common
- Soft `cumulative` constraint
  - At most N classes of some department are taught at the same time
  - N is a soft limit

Ill-defined problems

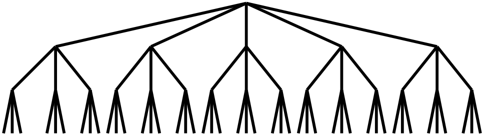
- Ill-defined problem
  - mistakes in the problem definition
- Mistake = contradiction of hard constraints
- Example
  - constraint propagation $\Rightarrow$ two teachers require the same classroom at the same time

Ill-defined problems

- Ill-defined problem
  - mistakes in the problem definition
- Mistake = contradiction of hard constraints
- Example
  - constraint propagation $\Rightarrow$ two teachers require the same classroom at the same time
- Mistakes must be removed
  from the problem definition to find a solution
- Solution: detection of mistakes
  - during search
  - during posting hard constraints

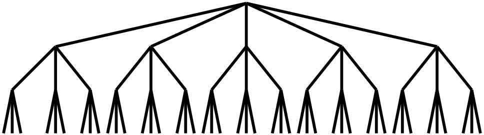
Tree Search

- Complete search
  - Depth First Search

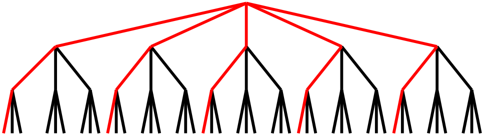


Tree Search

- Complete search
  - Depth First Search



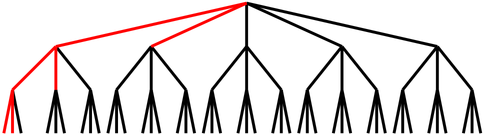
- Incomplete search: cutoff strategy
  - constrain some of the available resources
  - Depth Bounded Search



DBS(1)

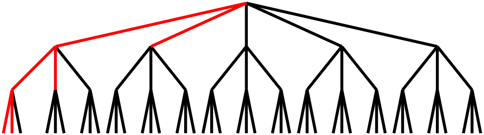
Limited Assignment Number Search

- LAN(2)

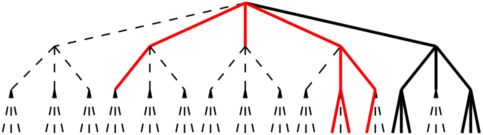


Limited Assignment Number Search

- LAN(2)



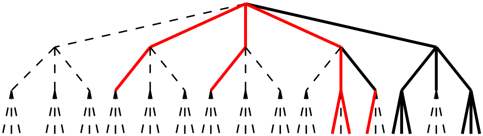
- Constraint propagation
  - values incompatible with the current partial assignment are removed from the domains
- LAN(3) + constraint propagation



  - do not waste time on „hard” variables
  - explore various parts of the search tree

Maximal Consistent Assignment

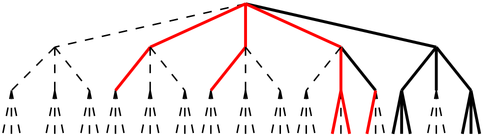
- Unassigned variables



  - second variable V_2 unassigned
  - some constraint(s) on V_2 remain(s) unsatisfied

Maximal Consistent Assignment

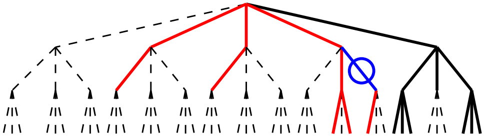
- Unassigned variables



  - second variable V_2 unassigned
  - some constraint(s) on V_2 remain(s) unsatisfied
- Consistent assignment
  - satisfy (at least) all constraints over assigned variables
- Maximal consistent assignment
  - consistent assignment of the largest cardinality
- Locally maximal consistent assignment
  - assignment which can not be extended to non-instantiated variable

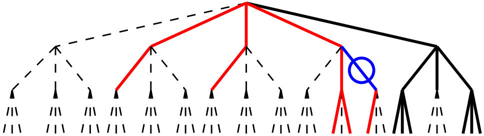
Restart for LAN Search

1. restart algorithm with different setting
   - unassigned variables first
   - untried values for unassigned values first
   - successful values for assigned variables



Restart for LAN Search

1. restart algorithm with different setting
   - unassigned variables first
   - untried values for unassigned values first
   - successful values for assigned variables



2. automated restart until the number of assigned variables increases
3. detection of possible problems over unassigned variables
   + hard to solve parts, weak propagation
4. continue with restart with the problem changes
   - problem redefinition does not introduce any changes in restart strategy

How to Discover Inconsistencies?

- Undesirable inconsistencies
- Mistakes in data input
  - minimize by user interface
  - types of mistakes
    - introduced during data entry
    - naturally included as a part of the problem

How to Discover Inconsistencies?

- Undesirable inconsistencies
- Mistakes in data input
  - minimize by user interface
  - types of mistakes
    - introduced during data entry
    - naturally included as a part of the problem
- Explanations
  - computationally expensive
  - not available in standalone constraint solvers
- During search
  - LAN search
    - user need some partial solution
    - user can change the problem
- Before search
  - manual process
  - automated process

Preconditions for Detection of Inconsistencies

- Detail (print) information about
  - posting each constraints
  - each value removal
- Post constraints in specific order
  - most complex first, most simple last
- Complex constraints
  - `disjoint2, cumulative`
  - detection of a conflict here is not easy
- Simple constraints
  - constraints over small set of variables
  - unary constraints over location or time

Towards Automated Process of Detection

1. Fail during constraint posting
2. Last posted constraint c_{LAST} failed
3. Variables in c_{LAST} are problem variables
4. Tracking of value removals for problem variables
5. Value removal was due to constraint c_{REMOVE}
6. Check consistency of c_{LAST} and c_{REMOVE}

In our problem, conflict between c_{LAST} and c_{REMOVE} was always source of a mistake.

Conclusion

Over-constrained, ill-defined problems

- solver for weighted soft constraints
- locally maximal consistent assignment during search
- detection of mistakes for standalone constraint solvers

Conclusion

Over-constrained, ill-defined problems

- solver for weighted soft constraints
- locally maximal consistent assignment during search
- detection of mistakes for standalone constraint solvers

Optimization

- multi-criteria optimization
- minimal perturbation problem