University Course Timetabling with Soft Constraints

Hana Rudová, Masaryk University
and
Keith Murray, Purdue University

PATAT 2002 Gent, Belgium

Class Timetabling at Purdue

Purdue University is a large (38,000 students) public university, noted for its programs in Engineering, Science, and Agriculture

Characteristics Affecting Timetables

- Most student curricula have a high proportion of required courses that must be taken in specific sequences
- Student scheduling procedures attempt to maximize the chances of students being placed in all of their required courses
  - Broad distribution of class times
  - Balanced filling of course sections
- However, little has been done to optimize the timetable based on student course requirements

Problem Statement

Assign times and rooms to all classes while minimizing the number of conflicts between requested courses and satisfying constraints on instructor and room availability.

- Room availability is major constraint for Purdue
  - Classrooms limited to force wide distribution class times
- Increased enrollments leave little excess room capacity
- Instructors want to control times they teach
- Standard time patterns
  - Example: 3 meetings x 50 minutes, 2 meetings x 75 minutes

Current Timetabling Process

Master class schedule constructed manually prior to enrollment

- Large lecture timetable – central scheduling office
  - 750 classes
  - 41 rooms
  - Many joint enrollments between classes (involves 29,000 students)
- Timetables for smaller classes – departmental timetablers
  - Approximately 70 disciplinary units
  - Up to 700 classes (approx. 100 average)
  - Up to 40 rooms

Solution Approach

- Desirable Decomposition
  - Large lectures vs. smaller classes within discipline
  - Smaller subproblems easier to solve (else 8,400 classes, 600 rooms)
  - Fits to political divisions of the University
- Flexiblity Necessary
  - Adaptable to 70+ problems with varying constraints
  - Allow changes after timetable published with with minimal disturbance
- Balancing Instructor Time Preferences Critical
  - Earlier attempt to automate timetabling unsatisfactory because solutions favored instructors who generated the most constraints
- Current Project: Large Lecture Problem

Constraint Logic Programming

- CLP programs are of the form:

```
solve(Variables) :- initialize_variables(Variables),
                    state_constraints(Variables),
                    search(Variables).
```

- `initialize variables` and `state constrains` define the model (declarative part of the solution)
- `search` defines the control part of the solution

Constraint Logic Programming (continued)

- Variables
  - Timetabling: time and classroom variables
- Constraints – hard constraints
  - Example: `disjunctive(Time1,Time2)` ensures different times for one instructor teaching two classes defined by `Time1` and `Time2`
- Search – finds assignment of values for all variables
  - Timetabling: problem of when (assign time variables) and where (assign classroom variables) classes must be taught

Constraint Logic Programming: Pros and Cons

- Advantage
  - Declarative model: easy extension for departamental timetables
- Problems
  - Preferential requirements
  - Solution of over-constrained problems
- Solution Proposed
  - Soft constraint solver
  - Search procedure for over-constrained problems

Soft Constraints

- A weighted CSP approach has been applied that considers weights/costs for each constraint and minimizes the weighted sum of unsatisfied constraints
  - Promotes more satisfactory solution without over-constraining problem
- Work included developing a new solver for soft constraints, implemented as an extension of the CLP(FD) library of SICStus Prolog
  - Allows use of existing hard constraints from CLP(FD) library and soft constraints from new solver
- Soft constraints are introduced via preference variables and preference propagation

Preference Variable

The `pref` unary constraint – assignment of initial preferences

Example: The unary soft constraint

```
pref( A, [7-5, 8-1, 10-0], _ )
```

creates a preference variable `A` with an initial domain containing the values `7`, `8`, and `10` with preferences `5`, `1`, and `0`, respectively

Other values are assumed to have infinite preference, indicating complete unsatisfaction

Preference Propagation

- Constraint Propagation – values are deleted from the domain of variable if a hard constraint is not satisfied
- Preference Propagation – cost of values in the domain of variable are increased if a soft constraint is not satisfied
- Example of preference propagation

  `soft_different(Start1, Start2, Cost)`

  Once one of preference variable `Start1` or `Start2` is instantiated to value `X`, inconsistency count for second variable should be increased by `Cost` for the value `X`

Aim

- All hard constraints in the CLP(FD) solver must be satisfied
  - Violation of hard constraints – new search method
- Optimization
  - Best inconsistency counts maintained by cost variables associated with each preference variable

    `pref(A, [7-5, 8-1, 10-0], Cost_Variable)`
  - All violations of soft constraints are stored in inconsistency counts

    $\Rightarrow$ aim is minimization of the sum of all cost variables

Model of Purdue Timetabling

- Time Variables
  - Preference variable (`pref` constraint) for the first meeting
  - Traditional domain variables for any additional meetings
  - Preferences = preferencial requirements for time of classes
- Classroom Variables
  - All meetings of one class must be in the same classroom

    $\Rightarrow$ one preference variable for classroom of class (`pref` constraint)
  - Preferences = preferencial requirements for suitable classroom
- Preferences on Class Non-overlapping
  - `soft_disjunctive` constraints on any two classes with common students

Optimization

- Minimize sum of cost variables for time preference variables
  - minimization of conflicts for students + optimization time preferences
- Minimize sum of cost variables for classroom preference variable
  - optimization of classroom preferences

Limited Assignment Number Search

- Iterative repair search: based on backtracking
  - Variable and value ordering heuristics improved iteratively
- LAN search algorithm developed sets a limit on number of times a value may be assigned to each variable
  - Complexity of search is linear with respect to number of variables, unlike full tree search in backtracking
  - However, search procedure is incomplete
- If limit is exceeded, variable is left unassigned and search continues with other variables
  - As a result, a partial assignment of variables is obtained together with the set of remaining unassigned variables

Partial Assignments

- Computation of partial assignments:
  - Avoids useless backtracks and returns partial results at any time
  - Allows handling of situations where the set of hard constraints over-constrains problem
  - Provides useful information for modifying search
  - Allows user to modify the problem statement
  - Allows user to relax constraints to eliminate contradictory requirements

Search in Timetabling Problem

1. LAN search over time variables
2. Search over classroom variables
   (a) Branch&Bound
   (b) If no solution is found within time limit
   $\Rightarrow$ replaced by LAN search

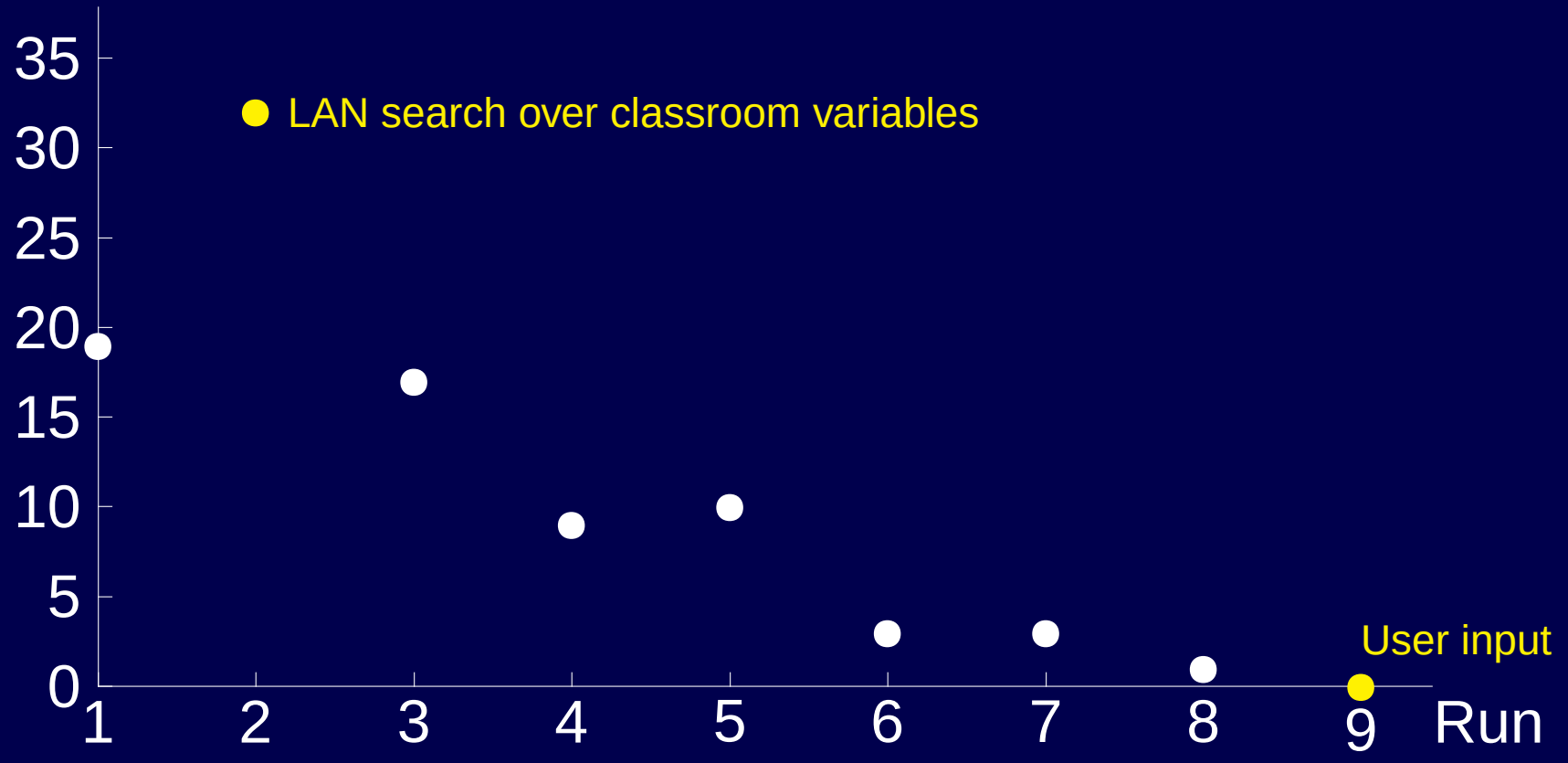
   Optimization over classroom variables is a second order requirement

Current Results

- Results (unpublished) for fall 2001 data using new variable ordering heuristic and LAN search for room labeling
  - Satisfied time gives the percentage of how many encouraged times for classes were selected – about 81 %
  - Unsatisfied time refers to the percentage of the discouraged times for classes which were selected – about 4 %
  - Student conflicts estimates the percentage of unsatisfied requirements for courses by students – about 2 %
  - Preferred classrooms measures the percentage of classes for which encouraged classrooms were selected – about 50 %

Unassigned Classes

Classes with either time or classroom variables that were not assigned during iterative runs of LAN search



Conclusions

- A large scale real problem was solved with soft constraints
  1. Over-constrained problem
  2. 98% of student course requests met
  3. 80% of preferential time requests met
- A new search algorithm for over-constrained and hard problems was proposed
- A new preference solver was implemented for soft constraints

Future Work

- Experiments with other data sets
- Initial sectioning
- Solutions for disciplinary problem
- Minimal perturbation problems
- Inclusion of distances between rooms via soft constraints
- Extentions of soft constraint solver
- Improvements of LAN search algorithm

Cost Variable

- An additional domain variable (cost variable) is maintained by the preference solver for each preference variable having the current best inconsistency count as its lower bound.

  `pref(A, [7-5, 8-1, 10-0], Cost_Variable )`

- Since the best inconsistency count is `0`, the initial lower bound of `Cost_Variable` is set to `0`. Any time the current best inconsistency count of the preference variable is increased, the lower bound of `Cost_Variable` is increased accordingly.
- Inconsistency counts can be increased either by preference propagation or by value removal.

Backtracking

- At each step, a value is assigned to a variable and propagated through the constraints into the domains of other variables
- If an assigned value is consistent with the constraints, the current partial solution is extended and the search continues to the next variable
- If none of the values in the domain of a variable results in a consistent solution, focus returns to the previous variable
- Variable and value ordering heuristics direct search towards promising parts of the search tree

Variable and Value Ordering

In subsequent iterations, variable and value ordering heuristics are developed as follows:

- Values of successfully assigned variables provide initial assignments in the subsequent iteration;
- Unsuccessful values of the unassigned variables are demoted in the ordering so that they will be tried last, in hope a suitable value will be among those not tried in the previous iteration;
- Information about unassigned variables is accumulated from all runs and variables with the greatest unassigned count are labeled first, as these variables may be more difficult to assign a value to than initially anticipated.