ITC 2007 – Examination Track: Geoffrey De Smet

Drools-solver

Introduction

Since late 2006 I've been working as a hobby on my local search framework, which uses the Drools rule engine for score calculation. This open source framework has been accepted by the Drools team as an experimental module, called **drools-solver**. Currently, I am still the only author, although the community interest is growing and contributions are welcome.

A local search algorithm and the Drools rule engine turn out to be a really nice combination, because:

- A rule engine such as Drools is **great for calculating the score** of a solution of a planning problem. It make it easy to add additional soft or hard constraints such as "a teacher shouldn't teach more then 7 hours a day". However it tends to be too complex to use to actually find new solutions.
- A local search algorithm is **great at finding new improving solutions** for a planning problem, without brute-forcing every possibility. However it needs to know the score of a solution and normally offers no support in calculating that score.

References

- Solver blog: http://blog.athico.com/search/label/solver
- Reference manual: http://users.telenet.be/geoffrey/tmp/solver/manual/html_single/
- Drools website: http://labs.jboss.com/drools/
- Source code: http://anonsvn.labs.jboss.com/labs/jbossrules/trunk/drools-solver/

ITC2007 examination track implementation

The examination track is one of the examples of drools-solver (others include N-Queens and the traveling tournament problem).

- **It's completely object orientated**, written in Java 6. There is a class for Exam, Period, Room, Student, PeriodHardConstraint, InstitutionalWeighting, ...
- **The score calculation is completely declarative**, written in DRL (drools rule language). Each score constraint is implemented as a score rule in few lines. Due to the forward chaining nature of those rules, they get the benefit of delta based score calculation for free.
- The ExaminationStartingSolutionInitializer initializes the Examination. It first sorts the Exams on student size (including coincidence and after period linked exams) and duration. Next, it iterates through the list and assigns each exam the best untaken spot. Coincidence exams are scheduled in the same period.
- Local search (actually tabu search) improves the solution found by the initializer:
  - There are 3 types of moves implemented: PeriodChangeMove, RoomChangeMove and ExamSwitchMove. Coincidence exams are always moved together to another period.
  - A percentage of all possible moves (0.2%) are randomly selected for evaluation. The 10 top moves of the last step (=iteration) are also selected.
  - A move that lead an already visited solution is tabu and is not accepted. A move that changes an exam which has been changed in the last 10 steps (=iterations) is also tabu.
  - The best accepted move is selected as the next step.

- The local search algorithm finishes once the time limit is reached.

Future improvements

- A forward-chained, stateful accumulate. The score rule “roomCapacityTooSmall” is getting backward chained, which seriously hurts performance (estimated at 50-90%).
- Aspiration for MoveTabuAccepter, UndoMoveTabuAccepter and PropertyTabuAccepter
- Tweak the move factory and selectors. Take advantage of the time gradient to select more moves near the end of the search.
- Tweak the dynamic lowering of the hard constraint weight for move decision.
- Multi-threading support to take advantage of multiple CPU's.
- A good SimulatedAnnealingAccepter: the current implementation is completely unoptimized.