Automated Timetabling using a SAT Encoding

Filip Marić

*Faculty of Mathematics
University of Belgrade

COST Action IC0901 Working Group 1 and Working Group 2 Meeting
and
Third Workshop on Formal and Automated Theorem Proving and Applications

SAT solvers

- tremendous progres in the last 15 years
- SAT solvers have become powerful enough to be used in many practical applications
- We argue that they can be used for automated timetabling for educational institutions.

SAT solvers

- tremendous progres in the last 15 years
- SAT solvers have become powerful enough to be used in many practical applications
- We argue that they can be used for automated timetabling for educational institutions.

Real-world applications

- real-world timetabling for educational institutions in Serbia
- successfully created 17 timetables for 4 different institutions
- 3 faculties (in 2 universities) and 1 high school in Belgrade

Approach

- Encode all timetable conditions by a propositional formula.
- Use SAT solver to search for a satisfying valuation.
- A satisfying valuation represents a valid timetable.

Decision vs optimization problem

The SAT problem is a decision problem, and timetabling is an optimization problem.

- Formulate very strict constraints so that each satisfying valuation is a good candidate for a final timetable.
- Incrementally add or remove soft constraints (SAT ascent/descent).

Problem description

Two different problem variants:

Basic course timetabling - assign given lessons to given time slots while obeying some given requirements.

Course timetabling with room allocation - additionally assign given lecture rooms to given lessons while obeying some given requirements.

Basic assumptions

- Per-week basis.
- Week is divided in days divided in equal-length time slots (periods).
- Lessons take one or several periods.
- Each lesson is taught by one or more teachers in one subject to one or more groups.
- Groups, teachers and lessons are known in advance.

Correctness requirements

- Each given lesson must be scheduled (exactly once).
- A teacher cannot teach two different subjects at the same time. It is possible that a teacher is required to teach the same subject to several different groups at the same time.
- A group cannot attend two or more different lessons at the same time.
- Only one teacher can occupy one room in one given period.

Comfort requirements

Very wide range of requirements can be formulated.

- Forbiden and requested teaching hours
- Group or teacher overlapping
- Teaching day duration
- Number of teaching days
- Consecutive teaching days
- Idle hours
- ...

Teaching time

- days - teaching days
- periods(d), $d \in$ days - periods in a day

Lessons - *tgsn*

All lessons are represented with a 4-tuple $tsgn$:

- t - Teacher
- s - Subject
- g - Group
- n - Number

Each lesson $tgsn$ has its duration($tgsn$).

Example

Teacher T teaches the subject S to the group G twice a week, once for 2 periods and once for 3 periods gives two lessons:
$TGS1$, duration$(TGS1) = 2$
$TGS2$, duration$(TGS2) = 3$

Encoding strategy

- A direct encoding is used.
- Basic and implied variables.
- Clauses that describe variable relationships.
- Clauses that describe constraints.

Basic variables

Begining of a lesson

x^{l}_{tsgndp} - *tsgn* begins in the period *p* of the day *d*.

Introduced for each lesson *tsgn*, day *d* and period *p*, such that:

$$\min(\text{periods}(d)) \leq p \leq \max(\text{periods}(d)) - \text{duration}(tsgn) + 1$$

The values of these variables uniquely determine the whole timetable

Basic variables

Begining of a lesson

x^{l}_{tsgndp} - *tsgn* begins in the period *p* of the day *d*.

Introduced for each lesson *tsgn*, day *d* and period *p*, such that:

$$\min(\text{periods}(d)) \leq p \leq \max(\text{periods}(d)) - \text{duration}(tsgn) + 1$$

The values of these variables uniquely determine the whole timetable

Implied variables (examples)

Duration of a lesson

x_{tsgndp} - $tsgn$ is held in the period p of the day d.

Introduced for each lesson $tsgn$, day d and period p.

Connecting the variables

$$x'_{tsgndp_1} \Rightarrow x_{tsgndp_2},$$

where

$$\min(\text{periods}(d)) \leq h_1 \leq \max(\text{periods}(d)) - \text{duration}(tsgn) + 1$$

$$h_1 \leq h_2 \leq h_1 + \text{duration}(tsgn) - 1.$$

$$x_{tsgndp_2} \Rightarrow \bigvee_{\substack{h_2 - \text{duration}(tsgn) + 1 \leq h_1 \leq h_2, \\ \min(\text{periods}(d)) \leq h_1 \leq \max(\text{periods}(d)) - \text{duration}(tsgn) + 1}} x'_{tsgndp_1},$$

where

$$\min(\text{periods}(d)) \leq h_2 \leq \max(\text{periods}(d)).$$

CNF conversion

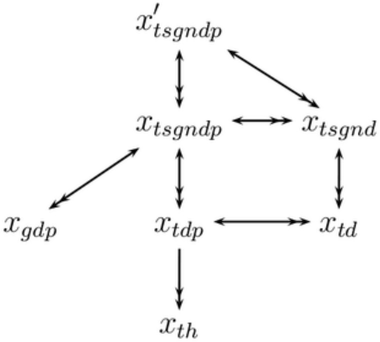
Implications

$$x \Rightarrow x_1 \vee \ldots \vee x_n$$

are trivially converted to clauses

$$\neg x \vee x_1 \vee \ldots \vee x_n.$$

Some other implied variables



Expressing constraints

- Introduced variables give a language suitable for expressing constraints.
- Constraints are given by additional clauses.
- Some auxiliry constructions (which are reduced to clauses) can be used to simplify specifications.

Auxiliary constructions

Cardinality constraints

$$\text{cardinality}(\{v_1, \ldots, v_k\}) \leq m$$

- at most m variables $v_1, \ldots, v_k$ are true.

Their encoding is well studied in the SAT literature (e.g., based on sequential counter circuts).

Single constraint

$\text{single}(\{v_1, \ldots, v_k\})$ - exactly one variable $v_1, \ldots, v_k$ is true.

Either reduced to cardinality constraints or trivially directly encoded.

Expressing constraints - examples

Example

Each lesson should be scheduled exactly once.

$$\text{single}(\{x'_{tsgndp} \mid d \in \text{days},\ p \in \text{periods}(d)\}).$$

The number of clauses can be reduced by:

$$\text{single}(\{x_{tsgnd} \mid d \in \text{days}\})$$
$$\text{single}(\{x'_{tsgndp} \mid p \in \text{periods}(d)\}),$$

for every $d \in$ days.

Expressing constraints - examples

Example

Each group can attend only a single class at a time.

$$\text{single}(\{x_{tsgndp} \mid tsgn \in \text{lessons}(g)\}),$$

for each $d \in$ days, and each $p \in$ periods(d).

Expressing constraints - examples

Example

Teacher t does not like to give lectures on Monday mornings.

$$\neg x_{t\_mon\_8}$$

Example

Group g must have lessons on Thursday 18h.

$$x_{g\_thu\_18}$$

Example

Teacher t likes to teach exactly two consecutive days.

$$\mathrm{cardinality}(\{x_{t\_mon}, x_{t\_tue}, x_{t\_wed}, x_{t\_thu}, x_{t\_fri}\}) = 2$$

$$x_{t\_mon} \Rightarrow x_{t\_tue}$$

$$x_{t\_tue} \Rightarrow x_{t\_mon} \vee x_{t\_wed}$$

$$x_{t\_wed} \Rightarrow x_{t\_tue} \vee x_{t\_thu}$$

$$x_{t\_thu} \Rightarrow x_{t\_wed} \vee x_{t\_fri}$$

$$x_{t\_fri} \Rightarrow x_{t\_thu}$$

Room allocation

Direct encoding -

- Introduce $x_{tsgndpr}$ variables.
- Becomes too complex for large number of rooms.

Cardinality based encoding -

- Do the timetabling in two phases:
  1. Perform only basic course timetabling, while ensuring that room allocation is possible.
  2. Perform the room allocation.

Cardinality based encoding - examples

How does one ensure that room allocation is possible?

Example

There are N rooms. Add the constraints:

$$\text{cardinality}(x_{tdp}) \leq N,$$

for each teacher t, day d and period p.

Things get more difficult when rooms are not equivalent (e.g., different capacities, computer labs), but this can still be managed.

SAT Optimization process

$$\text{cardinality}(\{\texttt{unsatisfiedsoftconstraints}\}) \leq k,$$

for different values of k. Different strategies:

- Increase k (SAT ascent)
- Decrease k (SAT descent)
- Binary search on k

Implementation

Custom input syntax (ASCII) for specifying constraints.

Example

```
days: mon tue wed thu fri
periods: 1-7
lessons:
   teacher1  group1, group2 subject1 2+1 room1
   teacher2  group1         subject2 3   room1, room2
   teacher2  group2         subject2 3   room1, room2
requirements:
   -teacher1_mon
   -group2_tue_7 | -group2_thu_1
```

Implementation

- Input specifications are converted to DIMACS by a simple encoder (written in C++).
- Formulae can be solved by any SAT solver.
- Models are easily back converted to timetables and displayed in HTML.

Solving times

Faculty of Mathematics -

- 80 teachers,
- 30 groups,
- two shifts,
- 2 buildings,
- 14 rooms,
- 97% room allocation in one shift,
- 12 periods per a day,
- cca. 650 lessons.

Solving time is around 5 minutes in average.

Solving times

Architectural high school -

- 85 teachers,
- 40 groups,
- two shifts,
- no need for room allocation,
- 14 periods per a day,
- cca. 1200 lessons.

Solving time is around 4 hours in average.

Implementation

- SAT solvers can be used for automated timetabling in small and medium sized educational institutions.
- Can handle a very wide range of requirements.
- Writing a SAT encoder is rather easy ($\leq$ 1000 lines of C++ code).
- There are many free SAT solvers available.
- Tecnhiques of SAT ascent/descent can be used to adapt the decision problem of SAT to an optimization problem of timetabling.
- Solving times showed not to be critical and they can probably be further reduced (e.g., use SMT instead of SAT).