TECHNISCHE UNIVERSITÄT DRESDEN

NPSOLVER

A SAT Based Solver For Optimization Problems

Norbert Manthey and Peter Steinke

Trento, 16.06.2012

Efficient SAT Solvers

During the last decade

- SAT solvers improved heavily
- There is an own research field
  - Search heuristics
  - Preprocessing and inprocessing
  - Parameter tuning
  - ...
- Each year, the performance of the tools increases
- The architecture turned parallel

Optimization Problems

Having a solution only is often not enough

- A solution should be
  - nice (nurse rostering)
  - small (size of a plan)
  - optimal (number of cargo trains per hour)

Optimization Problems

Having a solution only is often not enough

- A solution should be
  - nice (nurse rostering)
  - small (size of a plan)
  - optimal (number of cargo trains per hour)

- Searching the optimal solution is more complex

Optimization Problems

Having a solution only is often not enough

- A solution should be
  - nice (nurse rostering)
  - small (size of a plan)
  - optimal (number of cargo trains per hour)

- Searching the optimal solution is more complex
- Can we use advanced SAT technology efficiently?

Outline

Description of Instances

- There are hard constraints ...
  - Each train has to go on its route
  - The cost of these constraints is infinite
- ... and soft constraints
  - Having less trains carrying the same goods would be nice
  - Missing such a goal has a price (each)
- The overall cost has to be minimized

How to solve PBO

Problems are described for example as PB instance:

- $\sum_i w_i x_i \triangleright k$
- w_i and k are integers
- $\triangleright$ is a classical relational operators $=, >, <, \leq$ or $\geq$

How to solve PBO

Problems are described for example as PB instance:

- $\sum_i w_i x_i \triangleright k$
- w_i and k are integers
- $\triangleright$ is a classical relational operators $=, >, <, \leq$ or $\geq$

- Minimize the sum $\sum_i w_i x_i$

How to solve PBO

Problems are described for example as PB instance:

- $\sum_i w_i x_i \triangleright k$
- w_i and k are integers
- $\triangleright$ is a classical relational operators $=, >, <, \leq$ or $\geq$

- Minimize the sum $\sum_i w_i x_i$

Solving:

- Translate the instance to SAT and get a model J
- Evaluate $r = \sum_i w_i J(x_i)$
- Solver formula with new bound $r - 1$ until the formula is unsatisfiable

How to turn MaxSAT into PBO

MaxSAT:

- Clauses can have weights
- Minimize the weights of unsatisfied clauses

How to turn MaxSAT into PBO

MaxSAT:

- Clauses can have weights
- Minimize the weights of unsatisfied clauses

Translate into PBO:

- Clause C_i with weight w_i is turned into $C_i \vee x_i$
- We add to the current minimization $w_i \cdot x_i$
- Final result: minimize $\sum_i w_i x_i$ as in PBO

How to turn WBO into MaxSAT

Weighted Boolean Optimization (WBO):

- PB instances
- each PB constraint D_i can have a weight w_i

How to turn WBO into MaxSAT

Weighted Boolean Optimization (WBO):

- PB instances
- each PB constraint $\mathrm{D_i}$ can have a weight $\mathrm{w_i}$

Translate into PBO:

- Constraint $\mathrm{D_i}$ with weight $\mathrm{w_i}$ is turned into $\mathrm{D_i} \vee \mathrm{x_i}$
- We add to the current minimization $\mathrm{w_i} \cdot \mathrm{x_i}$

How to turn WBO into MaxSAT

Weighted Boolean Optimization (WBO):

- PB instances
- each PB constraint D_i can have a weight w_i

Translate into PBO:

- Constraint D_i with weight w_i is turned into $D_i \lor x_i$
- We add to the current minimization $w_i \cdot x_i$

Another translation into MaxSAT:

- Constraint D_i with weight w_i is translated into clauses $C_{i,j}$
- We turn each clause $C_{i,j}$ into $C_{i,j} \lor x_i$
- We add to the current minimization $w_i \cdot x_i$
- Note: the number of clauses is not changed

Outline

Reason for npSolver

Why do we implement npSolver?

- Its a good idea to use SAT solvers
- We can use any SAT solver (also parallel, default: glucose)
- PB instances are mixed with clauses and cardinality constraints
- Existing solvers (e.g. MiniSat+) do not support all features
- We can utilize incremental solving

Outline

Why should we translate PB into SAT?

We measured the distribution of constraints in the PB competition instances

- 90 % can be translated with clauses best (including BDD-path)
- Almost 10 % are general PB constraints
- There are very few cardinality constraints (for special encoding)
- There are only a few very large constraints

Why should we translate PB into SAT?

We measured the distribution of constraints in the PB competition instances

- 90 % can be translated with clauses best (including BDD-path)
- Almost 10 % are general PB constraints
- There are very few cardinality constraints (for special encoding)
- There are only a few very large constraints

Note:

- The measurement reflects all the constraints
- The distribution per instance can be different
- Currently, we support up to 64 bits

When to use which encoding?

What to do after the PB constraint have been read?

- Turn them into a $\leq$ type constraint
- Turn all weights into positive weights
- Determining the type of the constraint
  - trivial, at-most-one, at-most-k
  - general PB constraint with BDD, BDD-path or ADDERs
- Picking the right encoding
- Translating to SAT

When to use which encoding?

What to do after the PB constraint have been read?

- Picking the right encoding
- Translating to SAT

Encoding	AMO	2-product	Sorting NW	BDDs	Watch Dog	Adder NW
absolute	611859	19227	112253	22967061	517	567
relative	2.58 %	0.08 %	0.47 %	96.86 %	0.00 %	0.00 %

Details AMO

There are several ways to encode the at-most one constraint

- Pairwise encoding
- Sequential counters
- Log-encoding

TECHNISCHE UNIVERSITÄT DRESDEN

Details AMO

There are several ways to encode the at-most one constraint

- Pairwise encoding
- Sequential counters
- Log-encoding
- 2-product encoding (best asymptotic bound: $2\mathrm{n} + 4 \cdot \sqrt{\mathrm{n}} + \mathrm{O}(\sqrt[4]{\mathrm{n}})$)

Details AMO

There are several ways to encode the at-most one constraint

- Pairwise encoding
- Sequential counters
- Log-encoding
- 2-product encoding (best asymptotic bound: $2n + 4 \cdot \sqrt{n} + O(\sqrt[4]{n})$)
- Split-AMO
  - We split a bigger AMO and introduce fresh variables
  - $x_1 + \cdots + x_n \leq 1 \ \rightsquigarrow \ (y + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_i \leq 1) \ \bigwedge \ (\neg y + \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} x_i \leq 1)$
  - Produces $\sim 3n$ clauses, best for small n

Details AMO

There are several ways to encode the at-most one constraint

- Pairwise encoding
- Sequential counters
- Log-encoding
- 2-product encoding (best asymptotic bound: $2n + 4 \cdot \sqrt{n} + O(\sqrt[4]{n})$)
- Split-AMO
  - We split a bigger AMO and introduce fresh variables
  - $x_1 + \cdots + x_n \leq 1 \;\rightsquigarrow\; (y + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_i \leq 1) \;\bigwedge\; (\neg y + \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} x_i \leq 1)$
  - Produces $\sim 3n$ clauses, best for small n
  - Is there a reference for this?

Details on BDDs

BDDs can be understood as dynamic programming approach

- The weights of satisfied literals are summed up iteratively
- Per input variable the sum either stays or increases (ITE gate)
- If the bounds are reached, the translation can be stopped
  - The current sum is bigger than k
- Usually, only 2 clauses per node are needed for encoding a gate
- For incremental solving, only the last bound needs to be altered

Details on BDDs

BDDs can be understood as dynamic programming approach

- The weights of satisfied literals are summed up iteratively
- Per input variable the sum either stays or increases (ITE gate)
- If the bounds are reached, the translation can be stopped
  - The current sum is bigger than k
- Usually, only 2 clauses per node are needed for encoding a gate
- For incremental solving, only the last bound needs to be altered

- Note: we do not re-use gates yet among multiple PB constraints
- Note: if the path in the BDD to 0 are few, we encode the clauses

Outline

How to solve PBO

npSolver offers two main methods to search for an optimal solution

- Top-down search, by decreasing the bounds
- Binary search, always partitioning the remaining search space

How to solve PBO

npSolver offers two main methods to search for an optimal solution

- Top-down search, by decreasing the bounds
- Binary search, always partitioning the remaining search space

- We include an interface to incremental solvers via pipes

How to solve PBO

npSolver offers two main methods to search for an optimal solution

- Top-down search, by decreasing the bounds
- Binary search, always partitioning the remaining search space

- We include an interface to incremental solvers via pipes
- Surprisingly, non-incremental is best, both searches are equally well

How to solve PBO

npSolver offers two main methods to search for an optimal solution

- Top-down search, by decreasing the bounds
- Binary search, always partitioning the remaining search space

- We include an interface to incremental solvers via pipes
- Surprisingly, non-incremental is best, both searches are equally well

For MaxSAT, we decrease the number of k in the cardinality constraint

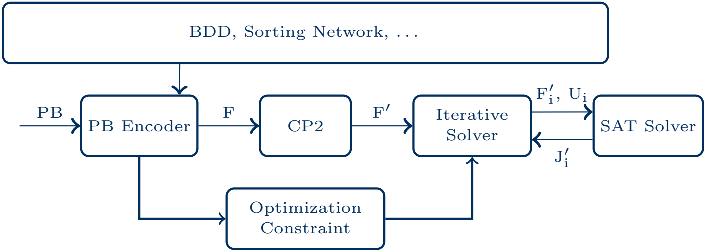
- Encoding the constraint needs several clauses
- Encoding a PB depends on k
- E.g. for weighted MaxSAT the constraint is huge
- By reducing k by some r, we can approximate and save clauses
- Final constraint: $\sum_i \lceil \frac{w_i}{r} \rceil < \lfloor \frac{k}{r} \rfloor$

Incremental Solving

- For the top-down approach, the solver can be kept
- Saving learned clauses should improve the search

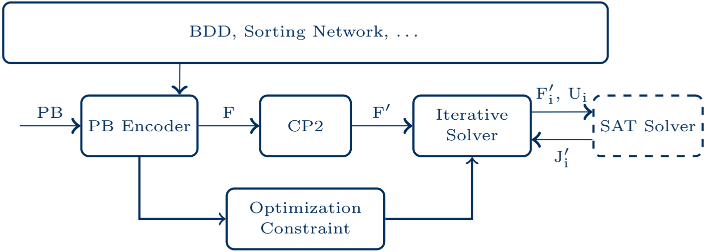
Incremental Solving

- For the top-down approach, the solver can be kept
- Saving learned clauses should improve the search



Incremental Solving

- For the top-down approach, the solver can be kept
- Saving learned clauses should improve the search
- In binary search, a new solver has to be created for failed formulas



Outline

How to use the tool

With npSolver, you can:

- Displays its parameters
- Easily encode PB into SAT (also MaxSAT and WBO)
- Gives statistics about the translation
- Use a SAT solver of your choice as solver
- Force the translation to a specific encoding

How to use the tool

With npSolver, you can:

- Displays its parameters
- Easily encode PB into SAT (also MaxSAT and WBO)
- Gives statistics about the translation
- Use a SAT solver of your choice as solver
- Force the translation to a specific encoding

Ongoing work includes

- Giving access to each encoding via a library
- Adding special cases for the "=" constraint (BDD,Adder)
- Support more encodings
- Improve modularity
- Furthermore: Optimization with (parallel) MaxSAT solvers?

Outline

Where can you find the tool?

`http://tools.computational-logic.org`

We (soon) provide

- Statically linked binaries
- The source code of the current version (under GPL 2)

- We will put updates and fixes online

Thanks for your attention

The solver is available at `http://tools.computational-logic.org`