Clause Sharing in Deterministic Parallel Maximum Satisfiability

Ruben Martins **Vasco Manquinho** Inês Lynce

IST/INESC-ID, Technical University of Lisbon, Portugal

RCRA 2012, Rome, Italy

Maximum Satisfiability

- Maximum Satisfiability (MaxSAT):
  - Optimization version of Boolean Satisfiability (SAT);
  - **Goal:** Given a propositional formula φ, find an assignment to problem variables that maximizes (minimizes) number of satisfied (unsatisfied) clauses in φ.

Maximum Satisfiability

- Maximum Satisfiability (MaxSAT):
  - Optimization version of Boolean Satisfiability (SAT);
  - **Goal:** Given a propositional formula φ, find an assignment to problem variables that maximizes (minimizes) number of satisfied (unsatisfied) clauses in φ.
- Partial MaxSAT
  - **Goal:** Given a propositional formula $\varphi = \varphi_h \bigcup \varphi_s$, find an assignment to problem variables such that all *hard* clauses in φ_h are satisfied, while minimizing the number of unsatisfied *soft* clauses in φ_s.

Maximum Satisfiability

- Maximum Satisfiability (MaxSAT):
  - Optimization version of Boolean Satisfiability (SAT);
  - **Goal:** Given a propositional formula φ, find an assignment to problem variables that maximizes (minimizes) number of satisfied (unsatisfied) clauses in φ.
- Partial MaxSAT
  - **Goal:** Given a propositional formula $\varphi = \varphi_h \bigcup \varphi_s$, find an assignment to problem variables such that all *hard* clauses in φ_h are satisfied, while minimizing the number of unsatisfied *soft* clauses in φ_s.

Maximum Satisfiability

- Main algorithmic approaches:
  - Branch and Bound
    - Extensive use of lower bounding procedures
    - Restrictive use of MaxSAT inference rules
  - Linear search on the number of unsatisfied clauses
    - Each time a new restriction is found, a new constraint is added that excludes solutions with higher cost
  - Unsatisfiability-based solvers
    - Iterative identification of unsatisfiable subformulas

Our focus is on the latter two approaches since these have been shown to be more effective in Industrial instances

Linear search on the number of unsatisfied clauses

Best: ∞

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example of MaxSAT formula; Hard clauses in blue; Soft in red

Linear search on the number of unsatisfied clauses

Best: ∞

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$

Add a relaxation variable to each soft clause; All clauses are now considered hard

Linear search on the number of unsatisfied clauses

Best: ∞

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$

Goal is to find an assignment that minimizes the number of relaxation variables assigned value 1

Linear search on the number of unsatisfied clauses

Best: 4

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$

Find a solution; Suppose a solution is found such that 4 relaxation variables are assigned value 1;

Linear search on the number of unsatisfied clauses

Best: 4

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$

$\sum_{i=1}^{7} r_i \leq 3$

Add new constraint that excludes solutions with equal or higher cost;

Linear search on the number of unsatisfied clauses

Best: 2

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$

$\sum_{i=1}^{7} r_i \leq 3$

Find another solution; Suppose a solution is found such that 2 relaxation variables are assigned value 1;

Linear search on the number of unsatisfied clauses

Best: 2

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$
$\sum_{i=1}^{7} r_i \leq 3$	$\sum_{i=1}^{7} r_i \leq 1$		

Add new constraint that excludes solutions with equal or higher cost;

Linear search on the number of unsatisfied clauses

Best: 2

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8 \vee r_2$	$x_6 \vee \neg x_8 \vee r_3$	$x_2 \vee x_4 \vee r_4$	$\neg x_4 \vee x_5 \vee r_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_7$
$\sum_{i=1}^{7} r_i \leq 3$	$\sum_{i=1}^{7} r_i \leq 1$		

Instance is now UNSAT; Optimal solution is to have two unsatisfied soft clauses

Unsatisfiability-based MaxSAT solvers

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example of MaxSAT formula; Hard clauses in blue; Soft in red;

Unsatisfiability-based MaxSAT solvers

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Formula is unsat; Get Unsatisfiable subformula (Unsat Core)

Unsatisfiability-based MaxSAT solvers

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_2$	$\neg x_4 \vee x_5 \vee r_3$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_4$

$\sum_{i=1}^{4} r_i \leq 1$

Add relaxation variables to soft clauses and AtMost1 constraint

Unsatisfiability-based MaxSAT solvers

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_2$	$\neg x_4 \vee x_5 \vee r_3$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_4$
$\sum_{i=1}^{4} r_i \leq 1$			

Formula is still unsat; Get another Unsat Core

Unsatisfiability-based MaxSAT solvers

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1 \vee r_1 \vee r_5$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_2$	$\neg x_4 \vee x_5 \vee r_3$
$x_7 \vee x_5$	$\neg x_7 \vee x_5 \vee r_6$	$\neg x_5 \vee x_3$	$\neg x_3 \vee r_4 \vee r_7$
$\sum_{i=1}^{4} r_i \leq 1$	$\sum_{i=5}^{7} r_i \leq 1$		

Add new relaxation variables to soft clauses in Unsat Core and AtMost1 constraint

Unsatisfiability-based MaxSAT solvers

$x_6 \vee x_2$ $\neg x_6 \vee x_2$ $\neg x_2 \vee x_1$ $\neg x_1 \vee r_1 \vee r_5$

$\neg x_6 \vee x_8$ $x_6 \vee \neg x_8$ $x_2 \vee x_4 \vee r_2$ $\neg x_4 \vee x_5 \vee r_3$

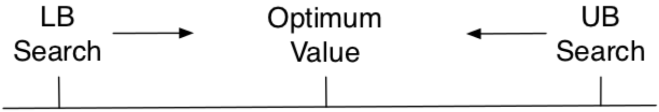
$x_7 \vee x_5$ $\neg x_7 \vee x_5 \vee r_6$ $\neg x_5 \vee x_3$ $\neg x_3 \vee r_4 \vee r_7$

$\sum_{i=1}^{4} r_i \leq 1$ $\sum_{i=5}^{7} r_i \leq 1$

Instance is now SAT; Algorithm Ends; Optimal solution is to have two unsatisfied soft clauses

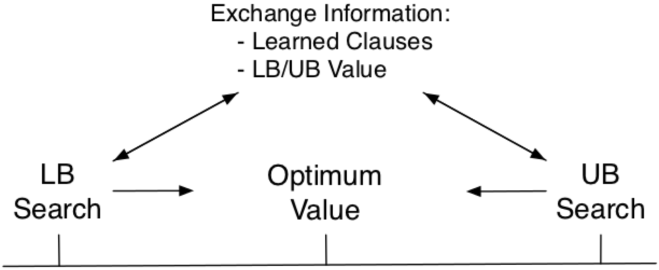
Parallel MaxSAT Solvers

- PWBO is a parallel MaxSAT solver based on having several threads running a portfolio of two orthogonal algorithms:
  - an unsatisfiability-based algorithm that searches on the lower bound of the optimal solution;
  - a classical linear search algorithm that searches on the upper bound.



Parallel MaxSAT Solvers

- PWBO is a parallel MaxSAT solver based on having several threads running a portfolio of two orthogonal algorithms:
  - an unsatisfiability-based algorithm that searches on the lower bound of the optimal solution;
  - a classical linear search algorithm that searches on the upper bound.



Parallel MaxSAT Solvers

- Shared Clause: a clause that is *shared* by a thread to be used in other threads;
- Imported Clause: a clause that is *imported* by a thread;

Parallel MaxSAT Solvers

- Shared Clause: a clause that is *shared* by a thread to be used in other threads;
- Imported Clause: a clause that is *imported* by a thread;

- Not all learned clauses should be shared/imported since it could lead to an exponential blow up in memory;
- Shared clauses can be imported or discarded by the receiving thread;

Parallel MaxSAT Solvers

- Shared Clause: a clause that is *shared* by a thread to be used in other threads;
- Imported Clause: a clause that is *imported* by a thread;

- Not all learned clauses should be shared/imported since it could lead to an exponential blow up in memory;
- Shared clauses can be imported or discarded by the receiving thread;

- **Question:** which learned clauses should be shared/imported by the different threads?

Clause Sharing Heuristics

- Static:
  - Learned clauses are shared/imported within a given cutoff.
- Dynamic:
  - Dynamic heuristics adjust the cutoff during the search.
- **Freezing:**
  - Shared clauses are temporarily frozen until they are expected to be useful.

Clause Sharing Heuristics (Static)

- Size:
  - The clause size is given by the number of literals;
  - Small clauses are expected to be more useful than larger clauses.
- Literal Block Distance (LBD):
  - The literal block distance corresponds to the number of different decision levels involved in a clause;
  - Clauses with small LBD are considered as more relevant.
- Random:
  - Randomly decide whether to share each learned clause with a given probability.

Clause Sharing Heuristics (Dynamic)

- The size of learned clauses tends to increase over time;
- Dynamic heuristics adjust the size of shared clauses during the search;

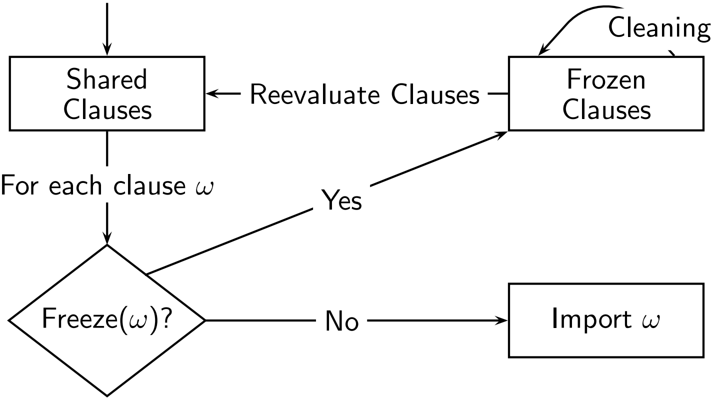
- Hamadi et al. proposed the following dynamic heuristic:
  - At every k conflicts the throughput of shared clauses is evaluated between each pair of threads $(t_i \rightarrow t_j)$;
  - If the sharing is small, the cutoff is dynamically increased;
  - If the sharing is large, the cutoff is dynamically reduced.

Clause Sharing Heuristics (Dynamic)

- The previous heuristic has been improved by Hamadi et al. by considering the quality of shared clauses:
  - A shared clause is said to have *quality* if at least half of its literals are active;
  - A literal is *active* if the variable's decision heuristic score is high, i.e. it is likely to be chosen as a decision variable in the near future;
  - If the quality is high then the increase (decrease) in the size limit of shared clauses will be larger (smaller).
- The reasoning behind this heuristic is that the information recently received from a thread t_i is qualitatively linked to the information which could be received from the same thread t_i in the near future.

Clause Sharing Heuristics (Freezing)

Freezing procedure for importing clauses shared by other threads



Clause Sharing Heuristics (Freezing)

The freezing heuristic:

- Considers the status of the shared clause ω in the context of the importing thread:
  - *Satisfied*: if at least one of its literals is satisfied;
  - *Unsatisfied*: if all of its literals are unsatisfied;
  - *Unit*: if all literals but one are unsatisfied and the remaining literal is unassigned;
  - *Unresolved*: if it is not satisfied, unsatisfied or unit.
- Freezes shared clauses ω that are not likely to be useful in the near future.

Clause Sharing Heuristics (Freezing)

- A satisfied clause is expected to be useful in the near future if:
  - It is not necessary to backtrack significantly to make the clause unit;
  - The number of unassigned literals that are not active literals is small;
- Unsatisfied clauses and unit clauses are always useful to the current search;
- An Unresolved clause is expected to be useful in the near future if:
  - The number of unassigned literals that are not active literals is small;

Clause Sharing Heuristics (Evaluation)

Question: How to properly evaluate all these clause sharing heuristics?

Clause Sharing Heuristics (Evaluation)

Question: How to properly evaluate all these clause sharing heuristics?

Observe that:

- Parallel solvers are non-deterministic due to cooperation between threads
- Cooperation is known to boost the performance of parallel solvers
- Variations might result from other factors than clause sharing procedures
- Therefore, a more stable environment is required for a fair evaluation

Clause Sharing Heuristics (Evaluation)

Question: How to properly evaluate all these clause sharing heuristics?

Observe that:

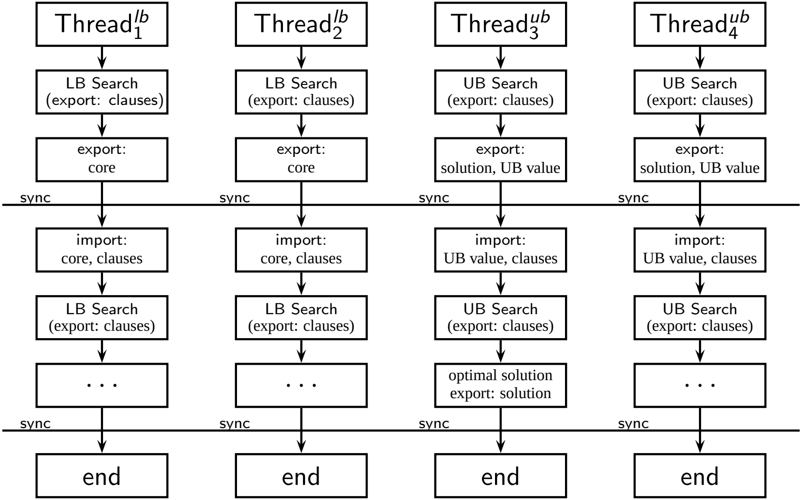
- Parallel solvers are non-deterministic due to cooperation between threads
- Cooperation is known to boost the performance of parallel solvers
- Variations might result from other factors than clause sharing procedures
- Therefore, a more stable environment is required for a fair evaluation

Proposed approach: test different clause sharing heuristics in a **deterministic** parallel MaxSAT solver

Deterministic Parallel MaxSAT Solver

- Cooperation between threads must be deterministic
- Introduction of synchronization points
- Information is only exchanged at synchronization points
- When a thread reaches a synchronization point, waits until **all** other threads reach the same point
- Only when all threads stop at the synchronization point the information exchange takes place

Deterministic Parallel MaxSAT Solver



Deterministic Parallel MaxSAT Solver

- The definition of synchronization points must be deterministic
- Example: Synchronize after k conflicts
- If k is small, number of synchronization points is large and threads are idle more often
- If k is large, there is little cooperation between threads
- For our experiments, we defined $k = 100$
- New ways of defining synchronization points are being tested

Experimental Results

- Benchmarks: partial MaxSAT instances from the industrial category of the MaxSAT Evaluation 2011:
  - Instances that took less than 60 seconds to be solved were not considered;
- AMD Opteron 6172 processors (2.1 GHz with 64 GB of RAM) running Fedora Core 13;
- Timeout: 1,800 seconds (wall clock time);
- Portfolio version of PWBO with 4 threads:
  - A deterministic version of PWBO was used;
  - Information is only exchanged at synchronization points (every 100 conflicts).

Experimental Results

Comparison of the different heuristics for sharing learned clauses

	Heuristic	*#Solved*	*Avg. #Clauses*	*Avg. Size*	*Time*	*Speedup*
	No sharing	137	–	–	32,188.57	1.00
Static	Random 30	134	10,140.22	128.21	27,394.46	1.18
Static	LBD 5	137	8,947.36	9.94	25,346.69	1.27
Static	Size 8	137	7,529.18	5.30	25,098.85	1.28
Static	Size 32	138	18,027.48	11.76	25,174.29	1.28
	Dynamic	138	13,296.28	7.33	24,218.84	1.33
	Freezing	140	16,228.53	11.01	21,611.21	1.49

- Randomly sharing clauses deteriorates the performance;
- LBD and size heuristics have similar speedups;
- Dynamic heuristic outperforms the static heuristics but is outperformed by the freezing heuristic.

Experimental Results

Non-deterministic vs. Deterministic version

Solver	#Solved	Time (s)	Avg. Idle CPU (%)	Speedup
Non-Deterministic	141	13,401.88	0	1.00
Deterministic	140	21,611.21	43.12	0.62

- Deterministic version is slower
- Number of solved instances is very similiar
- Large idle times should be decreased with other synchronization techniques

Conclusions

- Parallel MaxSAT solvers are now emerging:
  - Sharing learned clauses boosts the performance of the solver.
- Heuristics are used for sharing learned clauses:
  - Static, Dynamic and Freezing.
- Impact of sharing learned clauses in parallel MaxSAT:
  - Number of solved instances does not increase significantly;
  - Solving time is considerably reduced.
- The freezing heuristic outperforms all other heuristics both in solving time and number of solved instances.
- Deterministic parallel MaxSAT solver is slower but is still able to solve almost all instances solved by the non-deterministic version