Clause Sharing in Parallel MaxSAT

Ruben Martins **Vasco Manquinho** Inês Lynce

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

LION 2012, Paris

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

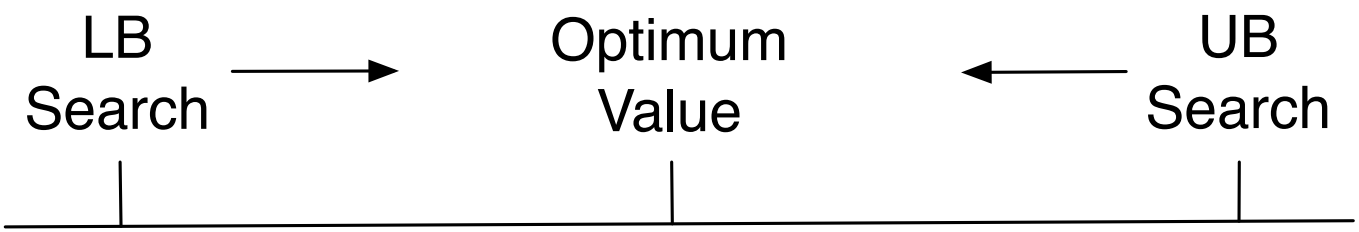
- Maximum Satisfiability (MaxSAT):
  - Optimization version of Boolean Satisfiability (SAT);
  - **Goal:** Given a propositional formula φ, find an assignment to problem variables that maximizes (minimizes) number of satisfied (unsatisfied) clauses in φ.
- Partial MaxSAT
  - **Goal:** Given a propositional 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.
- MaxSAT solvers are now very effective in practice;
- Multicore processors are becoming predominant;
- As a result, parallel MaxSAT solvers are emerging.

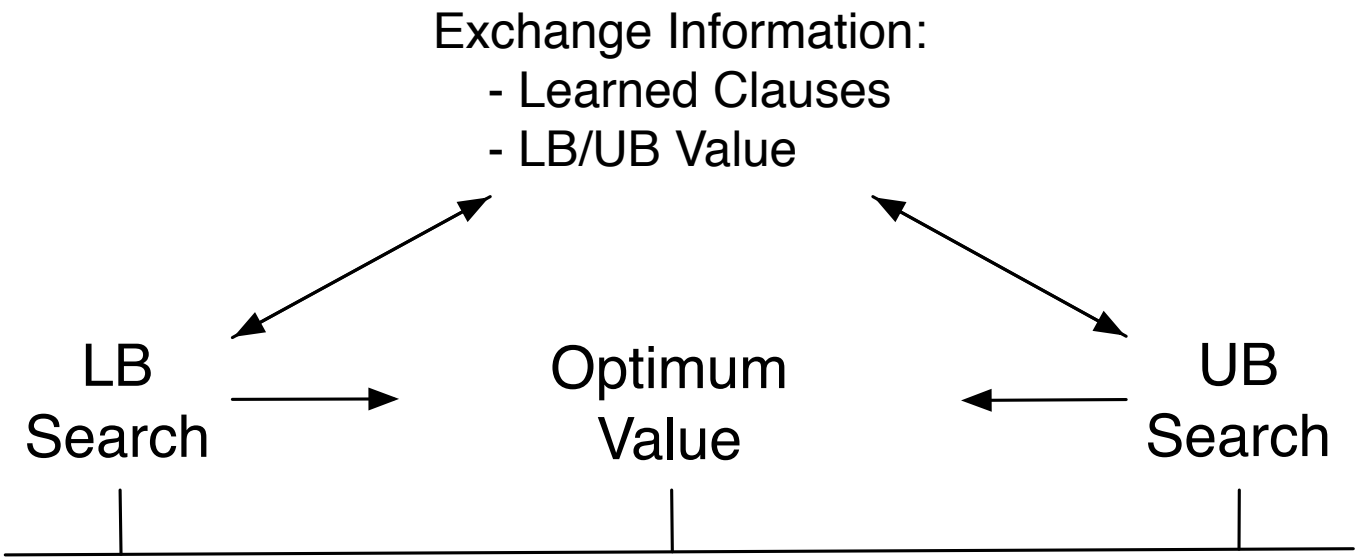
Parallel MaxSAT Solvers

- 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;

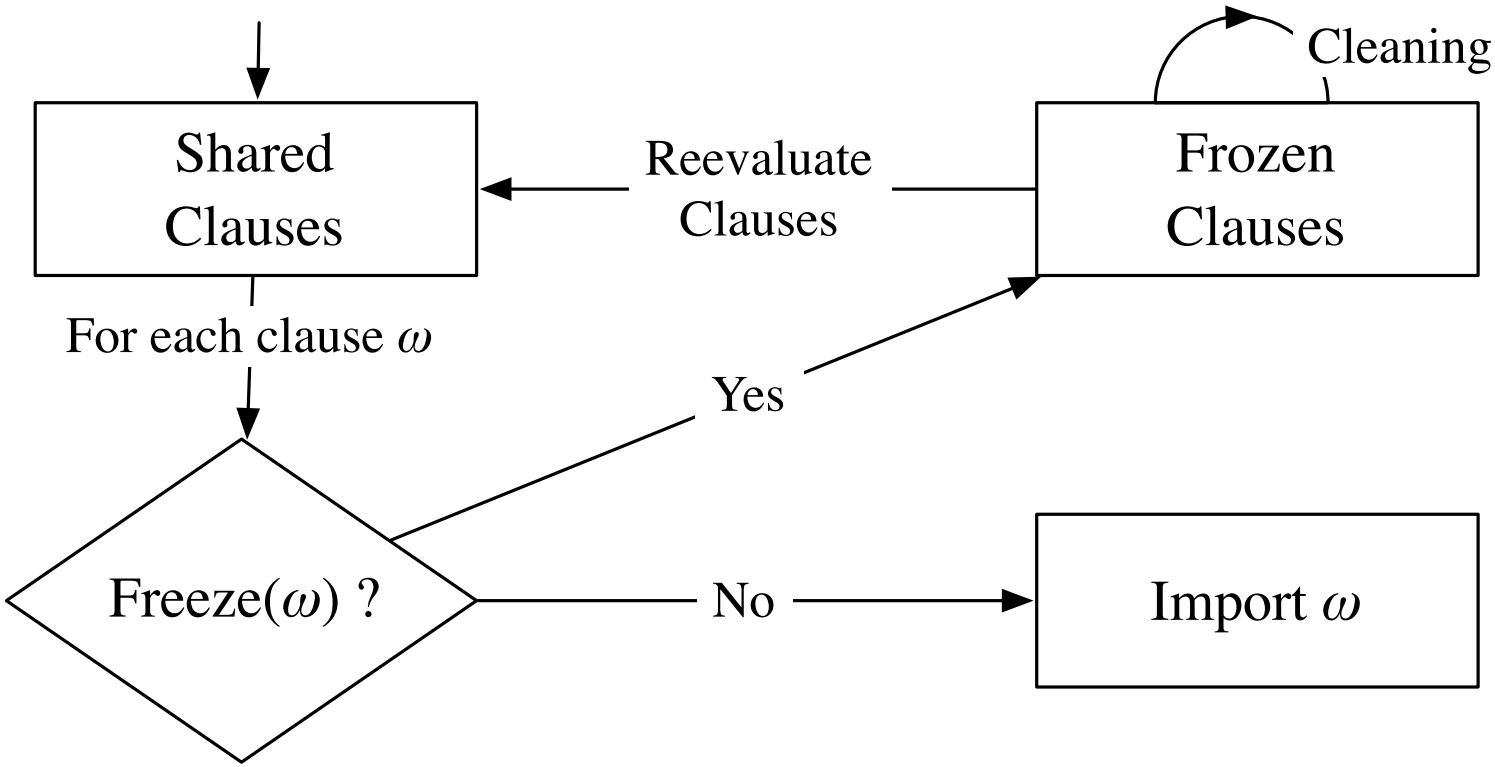
- Hamadi et al. proposed the following dynamic heuristic:
  - At every k conflicts the throughput of shared clauses is evaluated between each pair of threads ($t_i \rightarrow t_j$);
  - If the sharing is small, the cutoff is dynamically increased;
  - If the sharing is large, the cutoff is dynamically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

- $level$– current decision level, i.e. number of decisions since the *root* of the search tree until the current node
- $level_h(\omega)$– smallest decision level of the satisfied literals in ω;
- $unassignedLits(\omega)$– number of unassigned literals in ω;
- $activeLits(\omega)$– number of active literals in ω;

- ω is *satisfied*:
  - $(level - level_h(\omega) \leq 31)$;
  - $(unassignedLits(\omega) - activeLits(\omega) \leq 5)$;
  - If the above conditions are met ω is imported, otherwise it is frozen.

Clause Sharing Heuristics (Freezing)

- $level$– current decision level, i.e. number of decisions since the *root* of the search tree until the current node
- $level_h(\omega)$– smallest decision level of the satisfied literals in ω;
- $unassignedLits(\omega)$– number of unassigned literals in ω;
- $activeLits(\omega)$– number of active literals in ω;

- ω is *unsatisfied* or *unit*:
  - ω is always imported.
- ω is *unresolved*:
  - $(unassignedLits(\omega) - activeLits(\omega) \leq 5)$ then the clause is imported. Otherwise, it is frozen.

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.

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.