

# Control-based Clause Sharing in Parallel SAT Solving

[Youssef Hamadi](#), Microsoft Research, Cambridge

Said Jabbour, INRIA-Microsoft Research Joint Lab, Paris

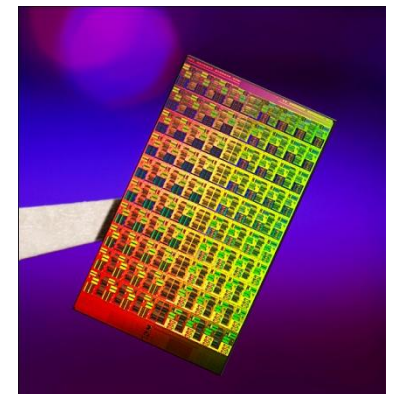
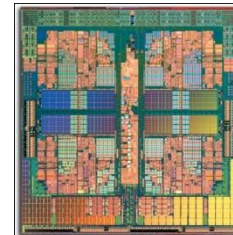
Lakhdar Sais, Université d'Artois, CRIL-CNRS, Lens

# Outline

- Parallelism, motivation and definitions
- Parallel tree-based search
- Clause sharing, benefit and problems
- Control-based clause sharing
- Summary, future work

# Motivation: technological

- **Thermal wall**: increases in processor clock frequency are slowing and in many cases frequencies are being decreased to reduce power consumption.
- **Scalability** through more computing units.
- **Moore's law**: the number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years.



# Motivation: algorithmic

- State of the art **sequential** algorithm looks difficult to improve (minor improvements, no orders of magnitude).
- SAT is applied to larger and **more ambitious problems** which cannot be solved in reasonable time.
  - ~90% of the industrial problems solved in less than 15min.
  - ~10% of the industrial problems solved in 1h.

# Definitions

- **Parallel system**: parallel algorithm + parallel architecture.
- **Scalability**: how well a parallel system takes advantage of increased computing resources.
  - Definitions:
    - Sequential runtime  $T_s$
    - Parallel runtime  $T_p$  (with  $p$  procs)
    - Parallel overhead  $T_o = pT_p - T_s$
    - Speedup  $S = T_s/T_p$
    - Efficiency  $E = S/p$
  - Typical objective: divide the sequential runtime by the number of resources, i.e.,  $E$  close to 1.

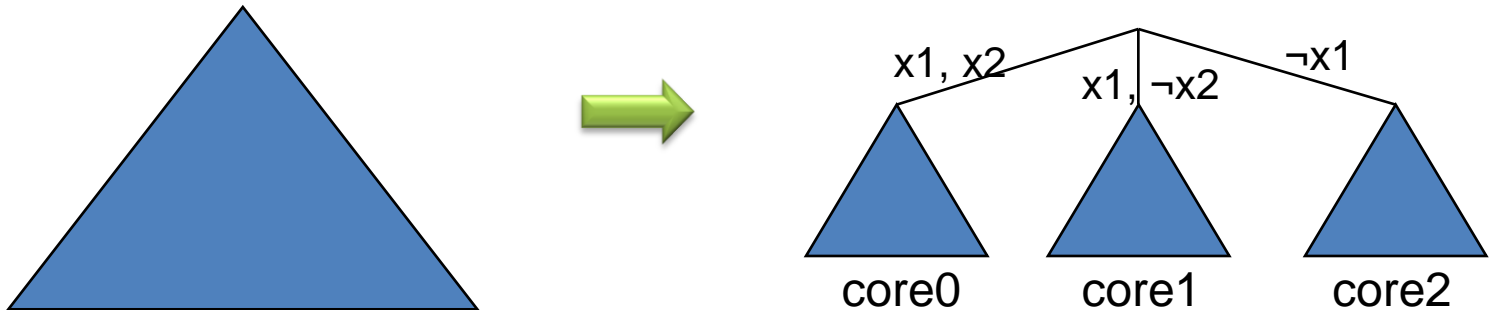
# Definitions

- **Knowledge**: information generated during the execution of a parallel algorithm.
- **Knowledge sharing**: mechanisms used to share the information. Many tradeoffs:
  - Cost of sharing:
    - Ramp up time
    - Communication overhead
  - Cost of not sharing:
    - Redundant work
    - Task starvation

# PARALLEL TREE-BASED SEARCH

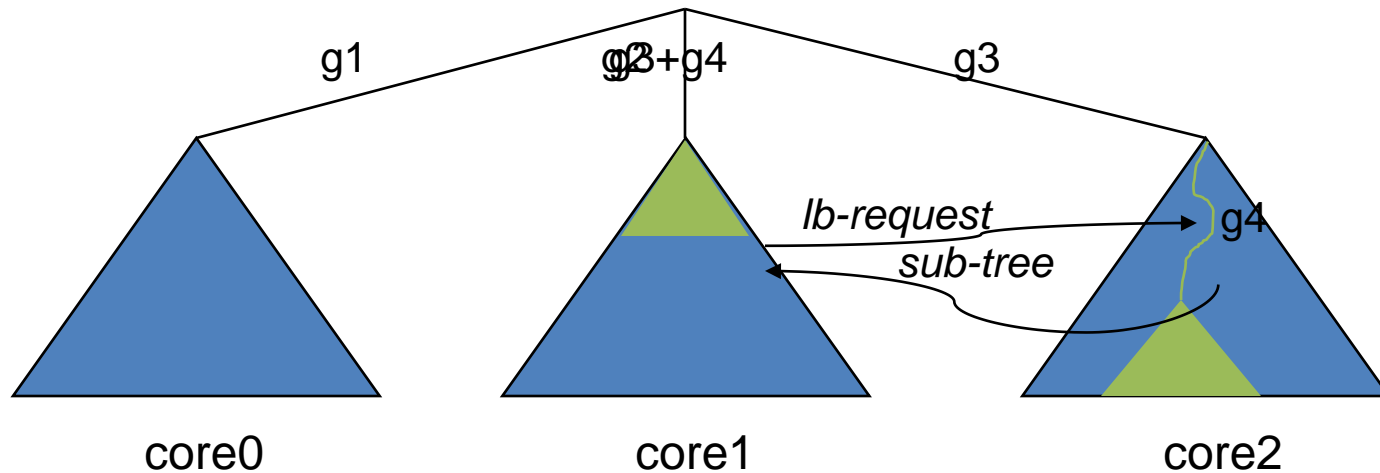
# Divide-and-conquer

- Principle: allocate independent sub-trees to different resources.
  - Use a set of constraints (a.k.a. **guiding paths**) to split the original search space.



- Problem: load imbalance

# Divide-and-conquer



- Knowledge Sharing:
  1. Load balancing
  2. Clause sharing

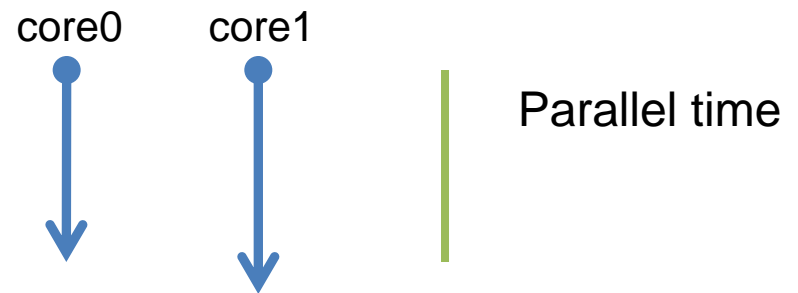
# Portfolio approach

- ManySAT [Hamadi, Jabbour, Sais 2008]
- Observation:
  - Modern SAT are sensible to their parameters, i.e., lack of robustness.
- ManySAT's principle: let several differentiated but related DPLLs **compete** and **cooperate** to be the first to solve a given instance. Tradeoffs:
  - Cover the space of search strategies
  - Exchange useful information

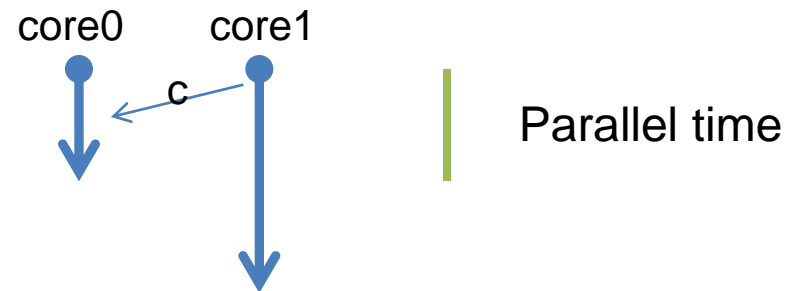
# Portfolio approach

- Knowledge sharing: clause sharing

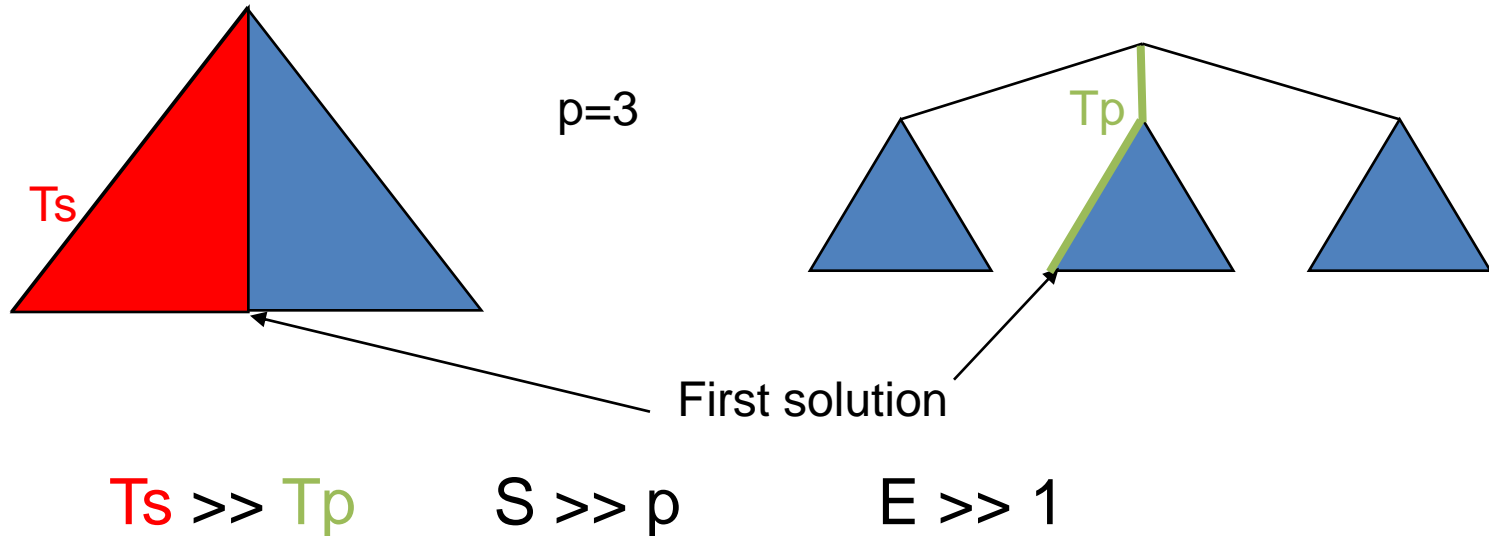
- Without: as good as the best



- With: better than the best



# Theoretical Performance



- “Speed-up anomalies in parallel tree search”, first reported identification circa 1975 [Pruul 88]
- [Rao et al. 93]: “... sequential DFS is sub-optimal...”
  - > Interleaved DFS (sequential) [Meseguer 97]
  - > Interleaved Backtracking in Distributed Constraint Networks [Hamadi 01]

# Practical Performance

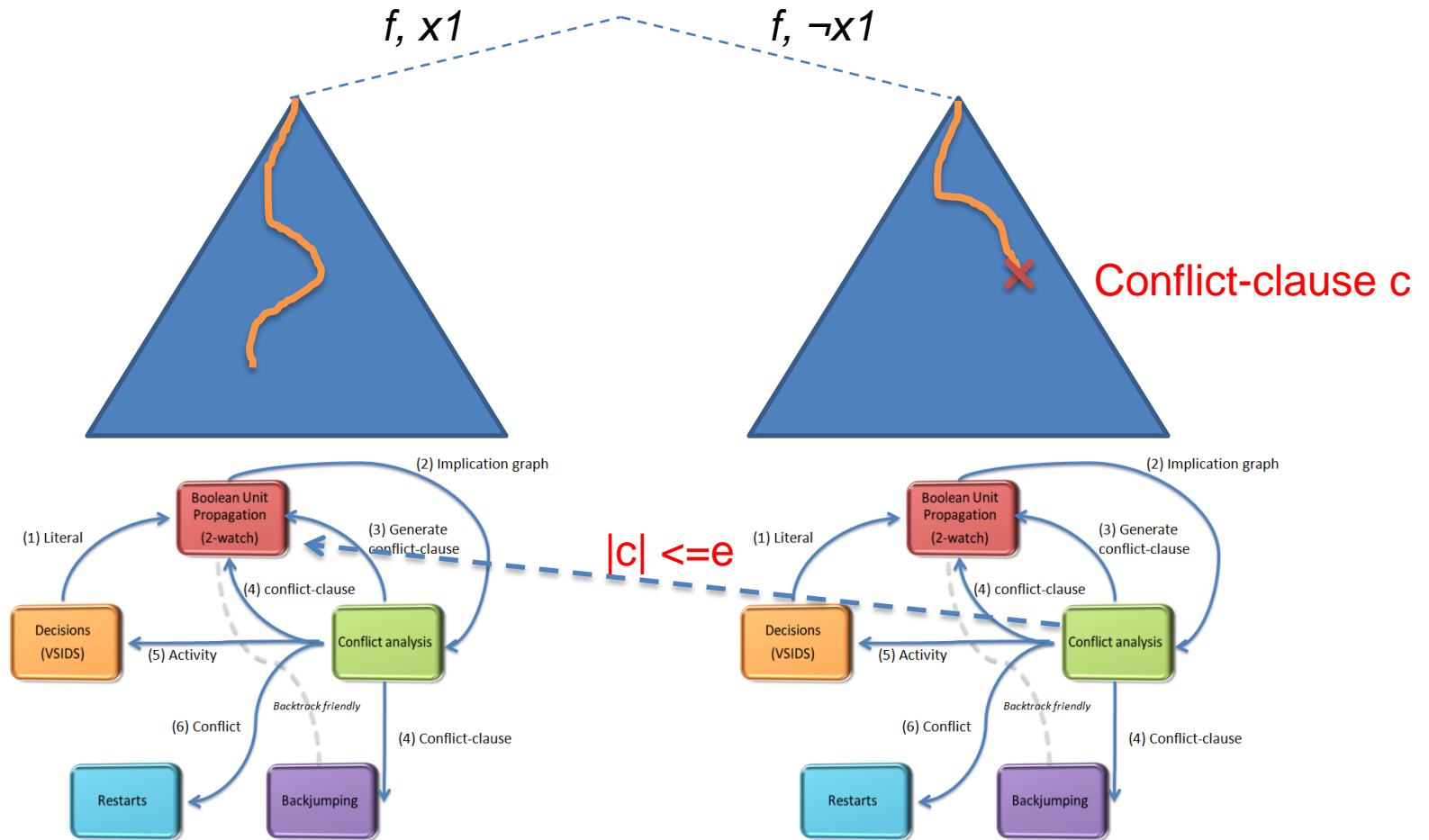
- SAT-Race 2008
  - 4 cores
  - 100 industrial problems
  - 900 seconds timeout
  - **Absolute speed-up** (vs. Minisat 2.1, best 2008 Sequential)



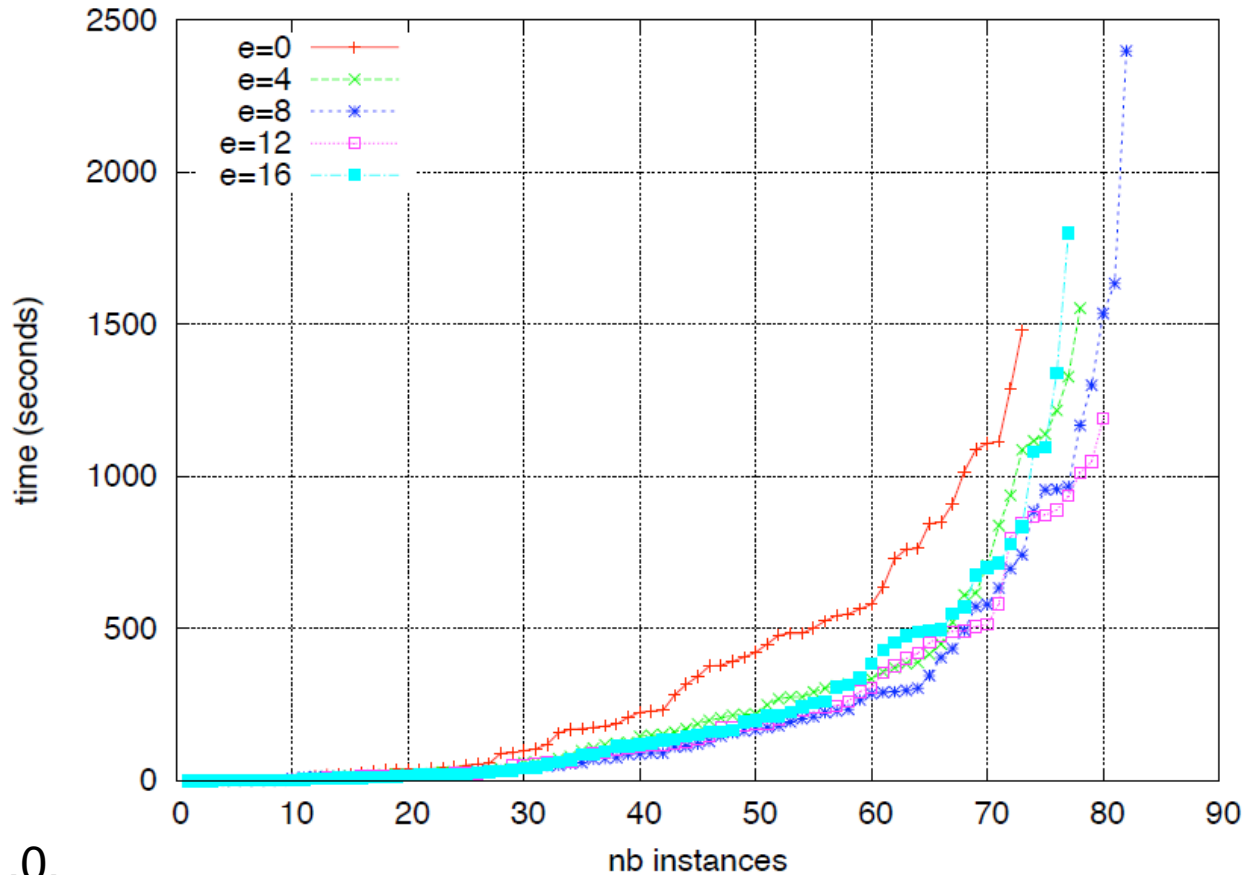
	ManySAT	pMinisat	MiraXT
#problems solved	<b>90</b>	85	73
Average speed-up	<b>6.02</b>	3.10	1.83
Minimal speed-up	<b>0.25</b>	0.34	0.04
Maximal speed-up	<b>250.17</b>	26.47	7.56
Average efficiency	<b>1.5</b>	0.77	0.45

# CLAUSE SHARING

# Standard Clause Sharing



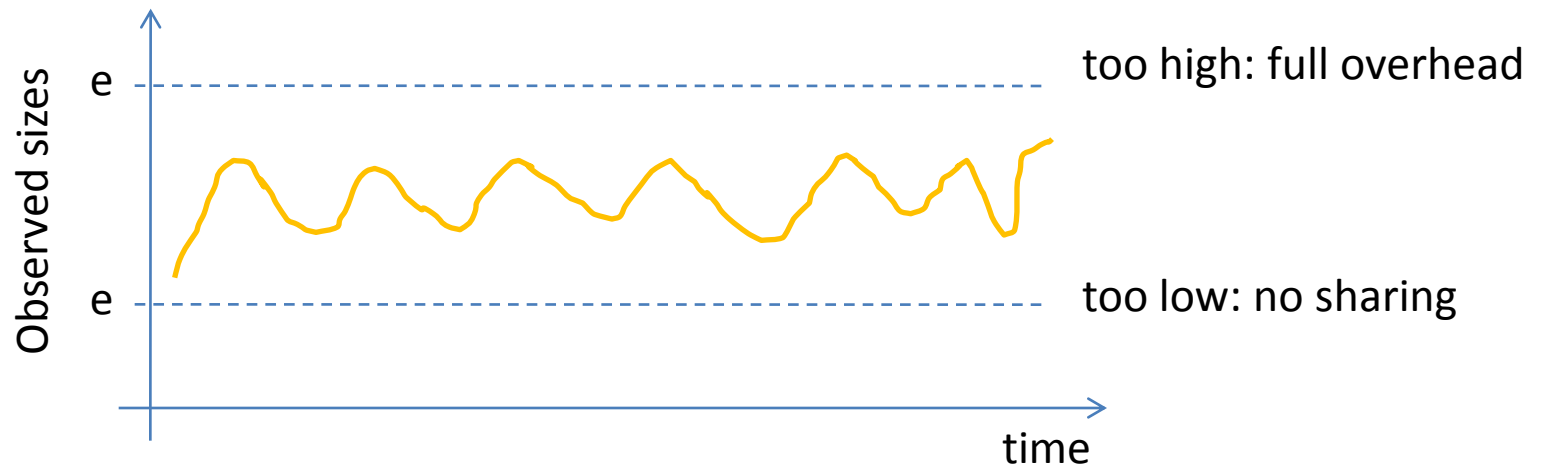
# Benefit



ManySAT 1.0,  
4 cores

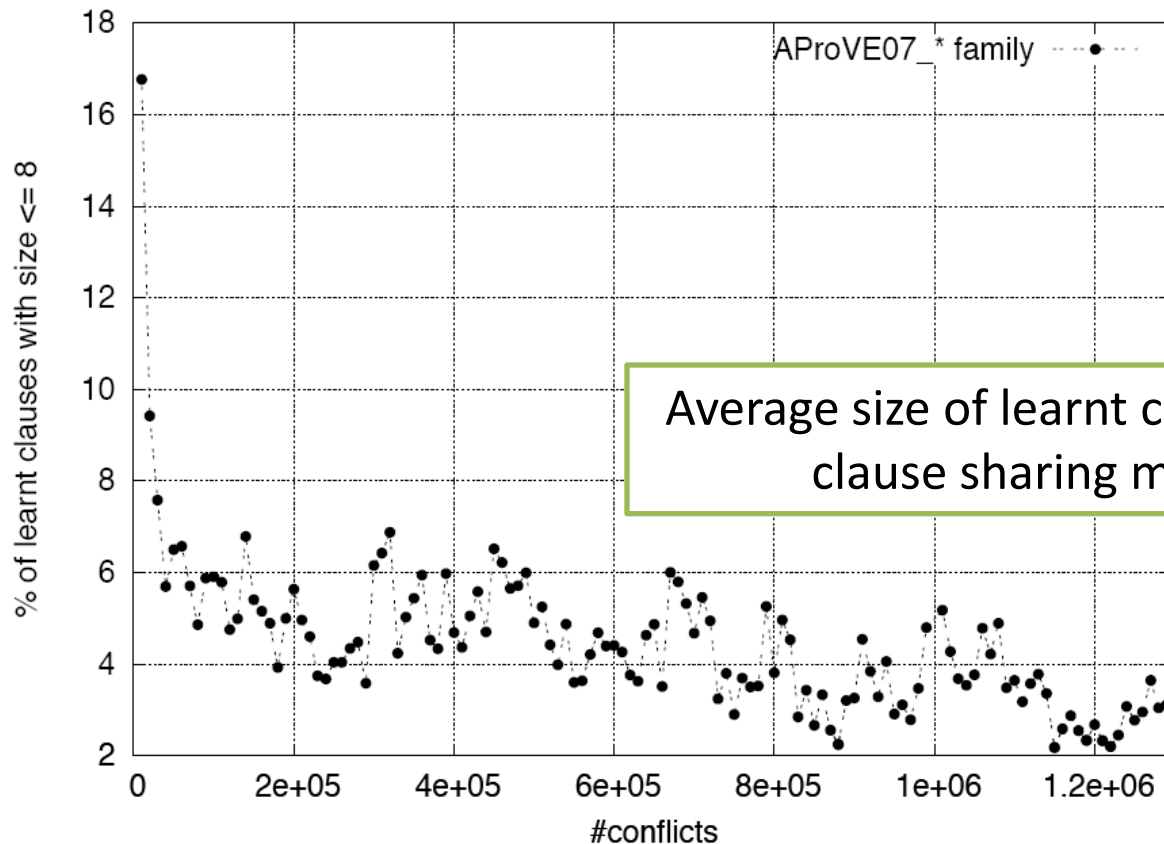
Figure 3. SAT-Race 2008: different limits for clause sharing

# Problem 1



# Problem 2

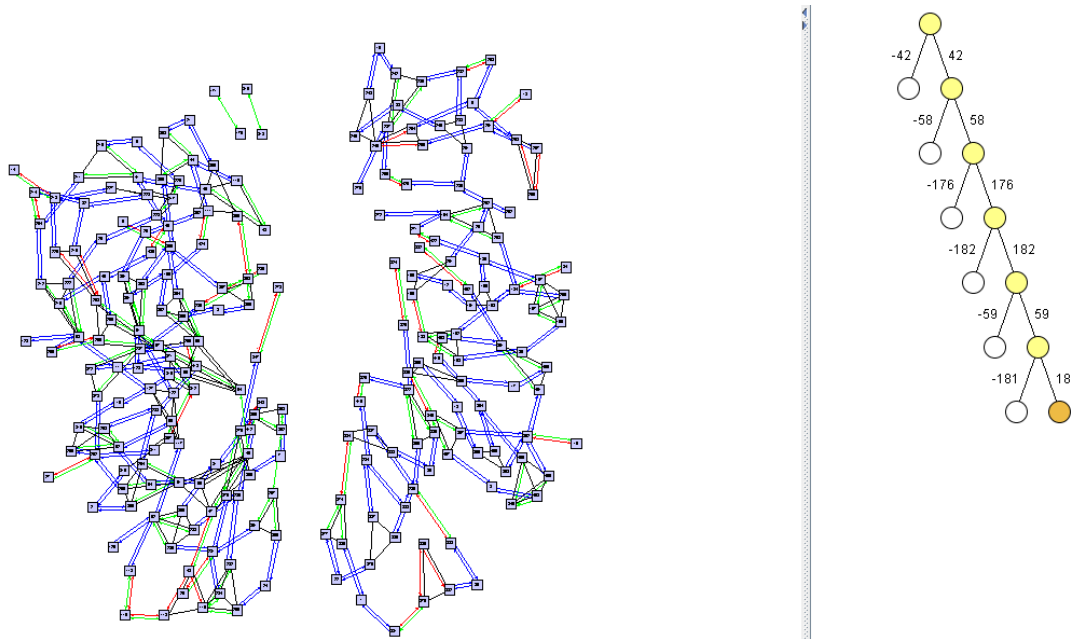
- Simple experiment with Minisat 2.0 (sequential)



Average size of learnt clauses is raising:  
clause sharing might halt.

# Problem 3

- Exchange between unrelated search efforts:

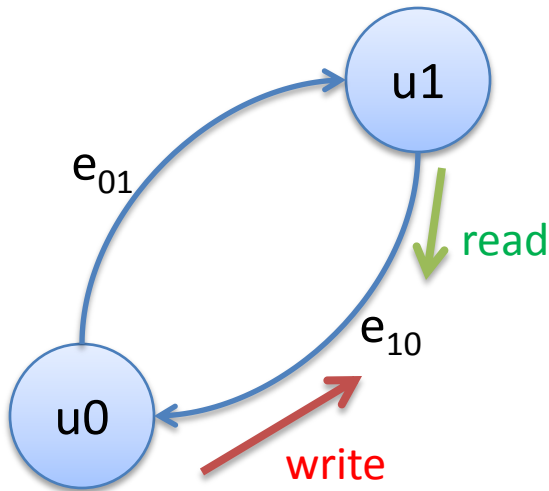


[DPVis, Sinz 05]

*Control-based Clause Sharing in Parallel SAT Solving*, Y. Hamadi, S. Jabbour, and L. Sais, Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09), July 2009, Pasadena, USA.

# **CONTROL-BASED CLAUSE SHARING IN PARALLEL SAT SOLVING**

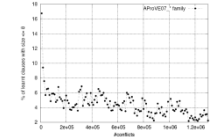
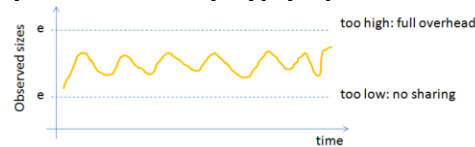
# Dynamic limits



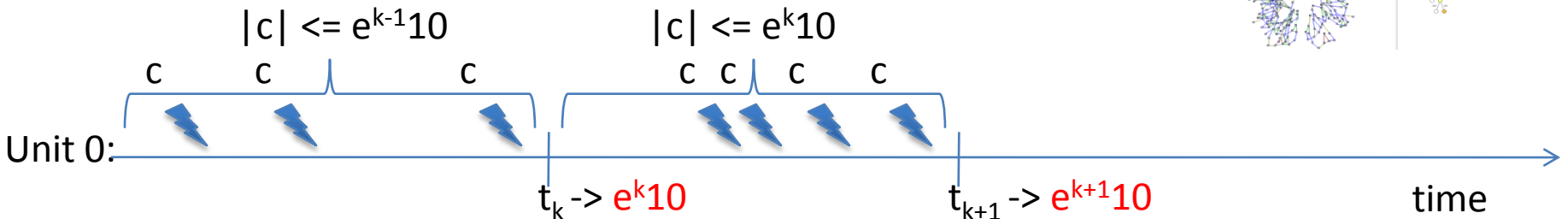
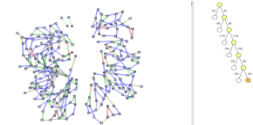
1. Pairwise size limits  $e_{ij}$  to control clause sharing from  $i$  to  $j$ .
2. Each unit performs (lock-free) periodic revisions of incoming limits.

Two objectives:

1. Maintain a **throughput T**. Solves problems (1), (2):



2. Maintain a **throughput T** of a given **Quality Q**. Solves (3):

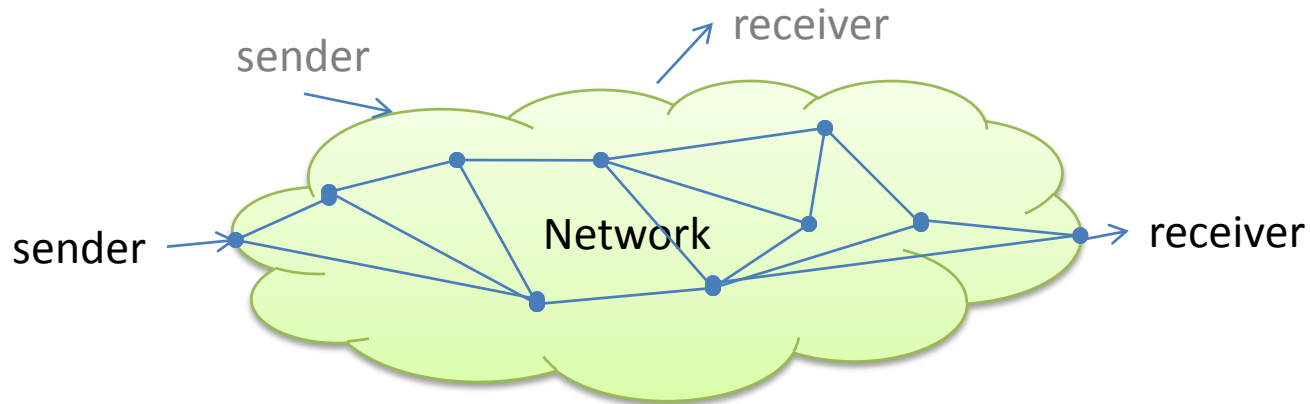


# Objective 1: Maintain a throughput $T$

- Throughput  $T$  is a number of foreign clauses received in each time interval
- Time interval =  $\alpha$  conflicts
- Typically,  $T = \alpha/c$
- Unit  $i$ , at step  $t_k$ :
  - $R_k$  is the number of foreign clauses received during  $t_{k-1}$
  - If  $R_k < T$ , uniform increase of  $e_{ji}^k$  limits
  - If  $R_k > T$ , uniform decrease of  $e_{ji}^k$  limits
- How do we update the limits?

# TCP Congestion Avoidance

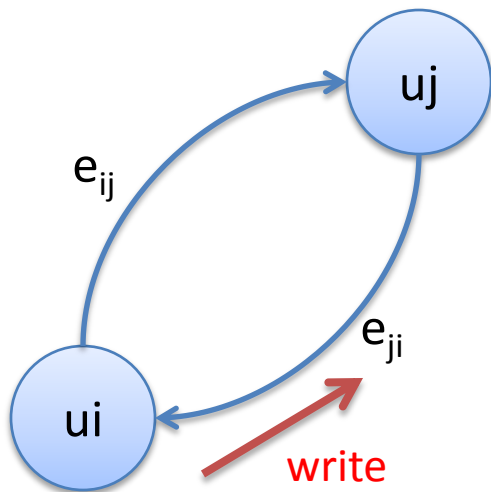
- Problem: guess the available bandwidth, i.e., find the correct communication rate  $w$ .



- Additive Increase Multiplicative Decrease (AIMD):
  - Slow increase as long as no packet loss:  $w = w + b/w$ 
    - i.e., probe for 'available' bandwidth.
  - Exponential decrease if a loss is encountered:  $w = w - a * w$ 
    - i.e., congestion: quick decrease for faster recovery.

# Additive Increase Multiplicative Decrease (AIMD)

- Clause sharing: an increase of the limits can generate a very large number of incoming clauses.
  - Slow increase, as long as  $T$  not met.
  - Exponential decrease, if  $T$  is met.



$$aimd_T(R_i^k) \{ \forall j | 0 \leq j < n, j \neq i$$

$$e_{j \rightarrow i}^{k+1} = \begin{cases} e_{j \rightarrow i}^k + \frac{b}{e_{j \rightarrow i}^k}, & \text{if } (|R_i^k| < T) \\ e_{j \rightarrow i}^k - a \times e_{j \rightarrow i}^k, & \text{if } (|R_i^k| > T) \end{cases}$$

# Objective 2: Maintain a throughput T of quality Q

- VSIDS heuristic: unassigned variables with the highest activity are related to the future evolution of the search process.

- Def.

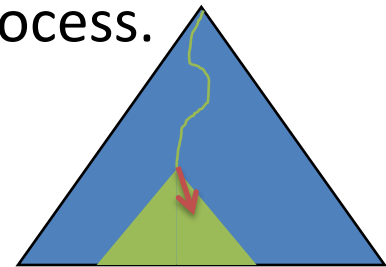
- Maximum VSIDS activity:  $A_i^{max}$
- Set of active literals of a foreign clause c:

$$\mathcal{L}_{\mathcal{A}_i}(c) = \{x/x \in c \text{ s.t. } \mathcal{A}_i(x) \geq \frac{A_i^{max}}{2}\}$$

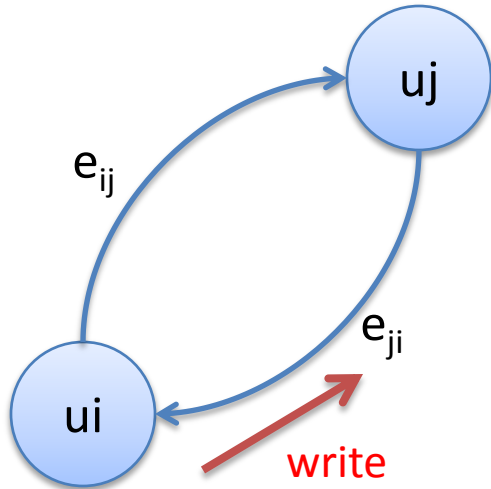
- Set of clauses received from j with at least Q active literals:

$$\mathcal{P}_{j \rightarrow i}^k = \{c/c \in \Delta_{j \rightarrow i}^k \text{ s.t. } |\mathcal{L}_{\mathcal{A}_i}(c)| \geq Q\}$$

- Quality of clauses received from j at step k:  $Q_{j \rightarrow i}^k = \frac{|\mathcal{P}_{j \rightarrow i}^k| + 1}{|\Delta_{j \rightarrow i}^k| + 1}$



# Maintain a throughput $T$ of quality $Q$



$$\begin{aligned}
 & \text{aimd}TQ(R_i^k) \{ \\
 & \quad \forall j | 0 \leq j < n, j \neq i \\
 & e_{j \rightarrow i}^{k+1} = \begin{cases} e_{j \rightarrow i}^k + \left(\frac{Q_{j \rightarrow i}^k}{100}\right) \times \frac{b}{e_{j \rightarrow i}^k}, \text{ if } (|R_i^k| < T) \\ e_{j \rightarrow i}^k - \left(1 - \frac{Q_{j \rightarrow i}^k}{100}\right) \times a \times e_{j \rightarrow i}^k, \text{ if } (|R_i^k| > T) \end{cases}
 \end{aligned}$$

- Increase/Decrease:
  - Favour units which give good quality clauses.

# EXPERIMENTS

# Parameters

- 4 cores
- $\alpha = 10000$  conflicts
- $T = \alpha/2$ , i.e., wants to receive 5000 foreign clauses
- $Q = |c|/3$
- Default AIMD parameters:

–  $a = 0.125$

–  $b = 8$

$$\begin{aligned}
 & \text{aimd}TQ(R_i^k) \{ \\
 & \quad \forall j | 0 \leq j < n, j \neq i \\
 & e_{j \rightarrow i}^{k+1} = \begin{cases} e_{j \rightarrow i}^k + \left(\frac{Q_{j \rightarrow i}^k}{100}\right) \times \frac{b}{e_{j \rightarrow i}^k}, & \text{if } (|R_i^k| < T) \\ e_{j \rightarrow i}^k - \left(1 - \frac{Q_{j \rightarrow i}^k}{100}\right) \times a \times e_{j \rightarrow i}^k, & \text{if } (|R_i^k| > T) \end{cases}
 \end{aligned}$$

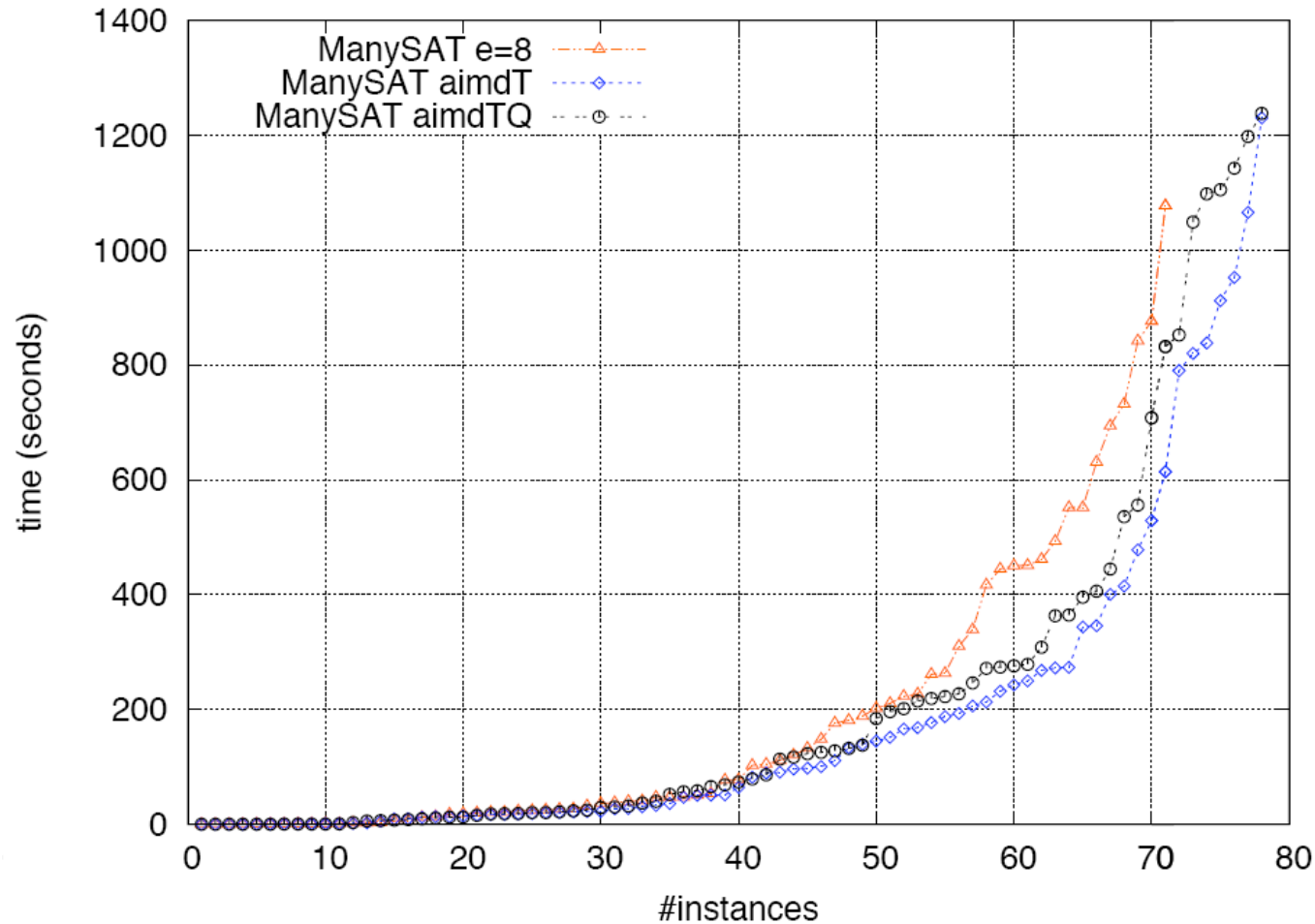
# Performance on Industrial problems

family/instance	#inst	ManySAT e=8		ManySAT aimdT			ManySAT aimdTQ		
		#Solved	time(s)	#Solved	time(s)	$\bar{e}$	#Solved	time(s)	$\bar{e}$
ibm_*	20	19	<b>204</b>	19	218	7	19	286	6
manol_*	10	10	<b>117</b>	10	<b>117</b>	8	10	205	7
mizh_*	10	6	762	7	746	6	<b>10</b>	<b>441</b>	5
post_*	10	9	325	9	<b>316</b>	7	9	375	7
velev_*	10	8	585	8	<b>448</b>	5	8	517	7
een_*	5	5	2	5	2	8	5	2	7
simon_*	5	5	111	5	84	10	5	<b>59</b>	9
bmc_*	4	4	7	4	7	7	4	<b>6</b>	9
gold_*	4	1	1160	1	<b>1103</b>	12	1	1159	12
anbul_*	3	2	742	<b>3</b>	<b>211</b>	11	3	689	11
babic_*	3	3	2	3	2	8	3	2	8
schup_*	3	3	129	3	<b>120</b>	5	3	160	5
fuhs_*	2	2	90	2	<b>59</b>	11	2	77	10
grieu_*	2	1	783	1	<b>750</b>	8	1	<b>750</b>	8
narain_*	2	1	786	1	<b>776</b>	8	1	792	8
palac_*	2	2	20	2	<b>8</b>	3	2	54	7
aloul-chnl11-13	1	0	1500	0	1500	11	0	1500	10
jarvi-eq-atree-9	1	1	70	1	69	25	1	<b>43</b>	17
marijn-philips	1	0	1500	<b>1</b>	1133	34	1	<b>1132</b>	29
maris-s03-gripper11	1	1	11	1	11	10	1	11	8
vange-col-abb313gpia-9-c	1	0	1500	0	1500	12	0	1500	12
Total/(average)	100	83	10406	86	9180	(10.28)	89	9760	(9.61)

Table 1: SAT-Race 2008, industrial problems

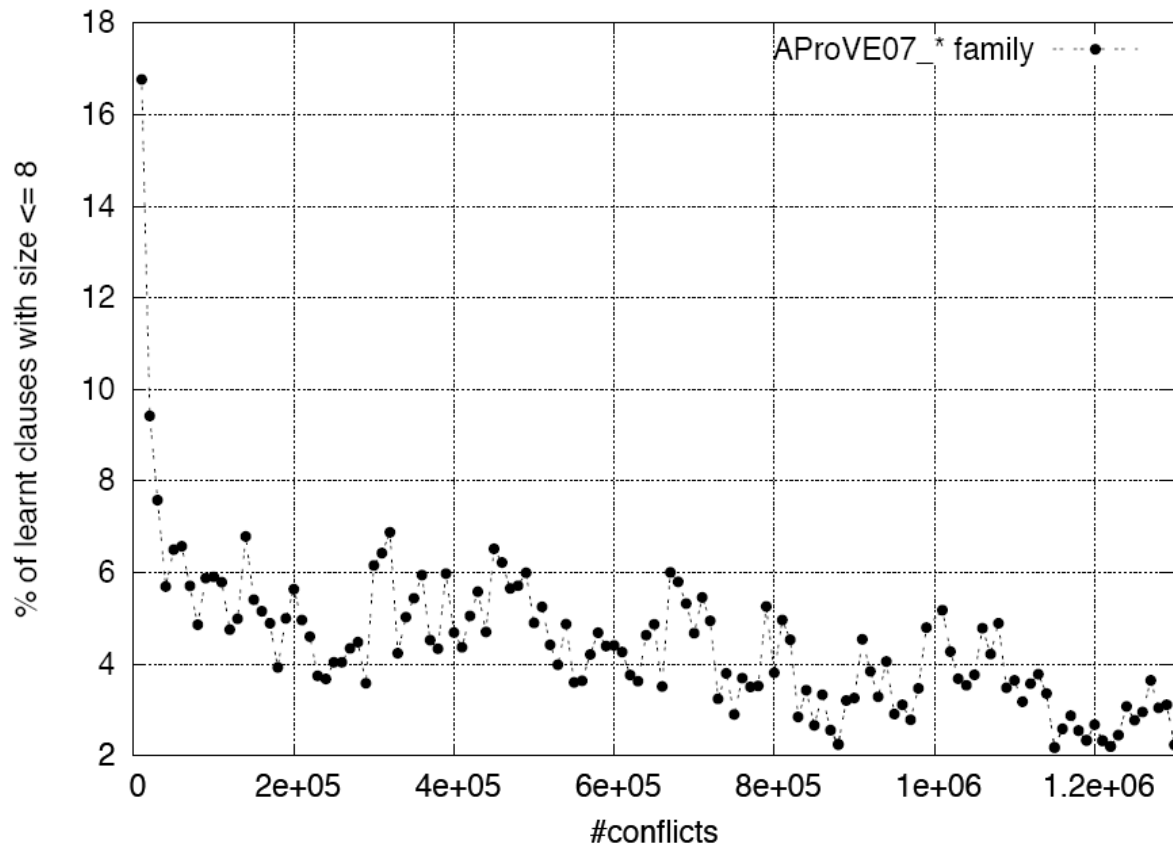


# Performance on Crafted problems (II)

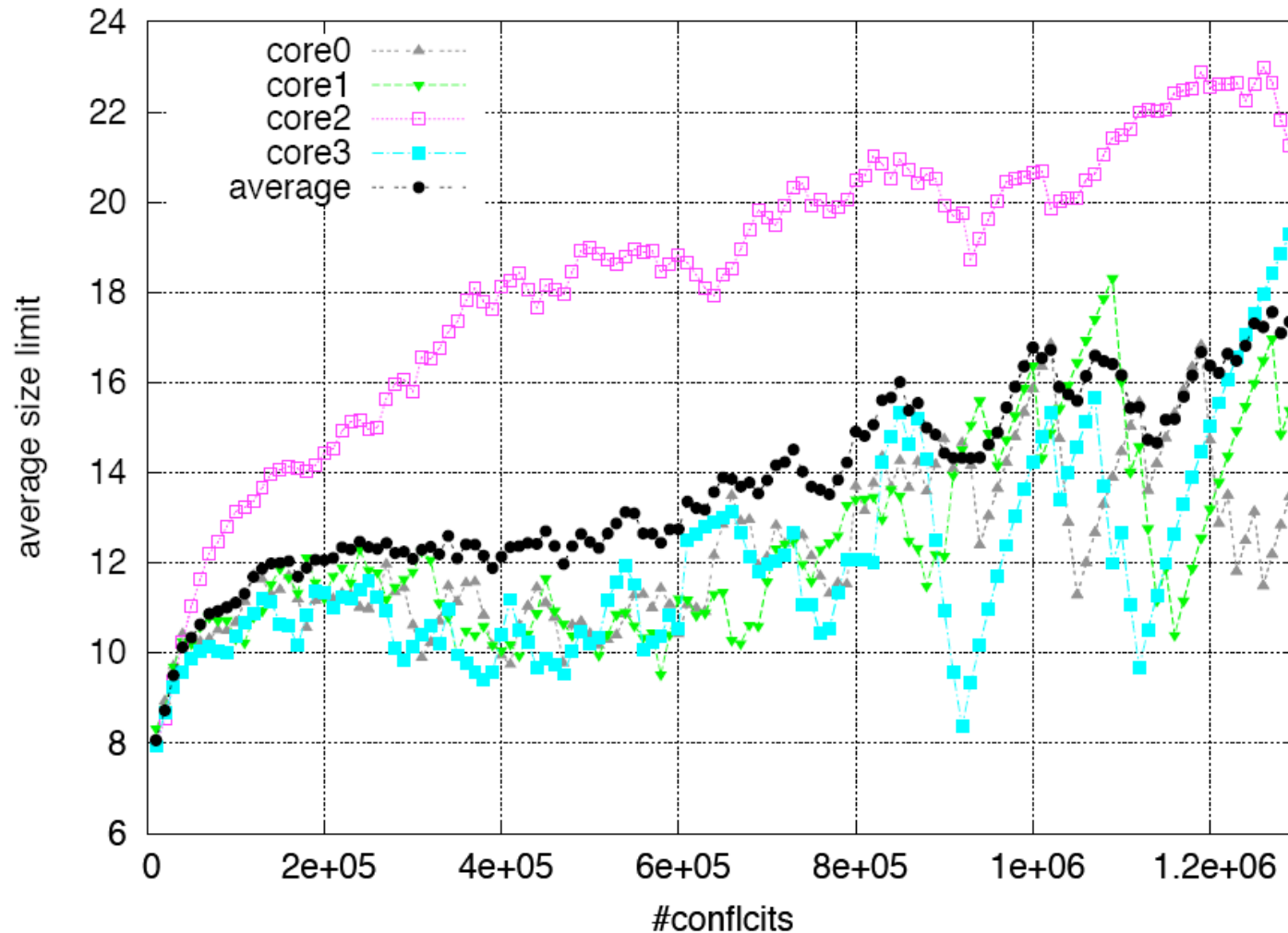


# Problems with clause sharing (2)

- Simple experiment with Minisat 2.0 (sequential)



# The dynamic of pairwise limits



# Summary

- Knowledge sharing is crucial:
  - Better than the best
  - There are overheads
- Problems fixed-size limit policy
  - Too low/high
  - Raising size of learnt clauses
  - Unrelated search efforts
- Control theory to adapt the rate of KS to a given instance:
  - Automatically 'disconnect' unrelated search efforts
  - Automatically reinforce exchanges between related ones
  - Solves ManySAT's tradeoffs

# Related and future work

- SMT Solver, Z3 2.0 (//Z3)
  - A Concurrent Portfolio Approach to SMT Solving, C. M. Wintersteiger, Y. Hamadi, L. M. de Moura CAV 2009: 715-720
- Very large scale parallel SAT solving
  - Experiments with Massively Parallel Constraint Solving, L. Bordeaux, Y. Hamadi, and H. Samulowitz, Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09), July 2009, Pasadena, USA.
- Future work
  - Implement in a divide-and-conquer approach
  - In other formalisms (QBF?)
  - Better definitions for the *Quality* of a clause
  - Distributed scenarios with heterogeneous communication costs