Revisiting Clause Exchange in Parallel SAT Solving

Gilles Audemard, Benoît Hoessen, Saïd Jabbour,
Jean-Marie Lagniez, Cédric Piette

Université Lille-Nord de France
CRIL - CNRS UMR 8188
Artois, F-62307 Lens
{*audemard,hoessen,jabbour,piette*}@cril.fr

Institute for Formal Models and Verification
Johannes Kepler University,
AT-4040 Linz, Austria
{*jean-marie.lagniez*}@jku.at

June 18, 2012

This work has been supported by CNRS and OSEO, under the ISI project “Pajero”.

SAT framework

SAT framework

- Lots of solvers are of type *Conflict driven clause learning* (CDCL)

SAT framework

- Lots of solvers are of type *Conflict driven clause learning* (CDCL)
- CDCL creates one new clause at each conflict

SAT framework

- Lots of solvers are of type *Conflict driven clause learning* (CDCL)
- CDCL creates one new clause at each conflict
- Restarts are quite frequent

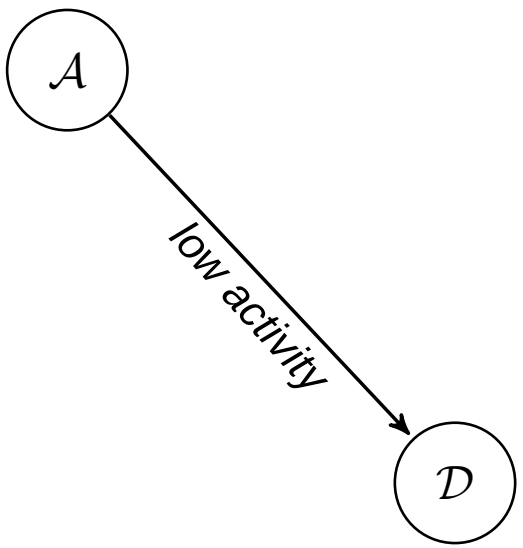
SAT framework

- Lots of solvers are of type *Conflict driven clause learning* (CDCL)
- CDCL creates one new clause at each conflict
- Restarts are quite frequent
- *Literal block distance* (*lbd*) provides a qualitative measure about learnt clauses (glucose)

Freezing learnt clauses

Classical learnt clause management

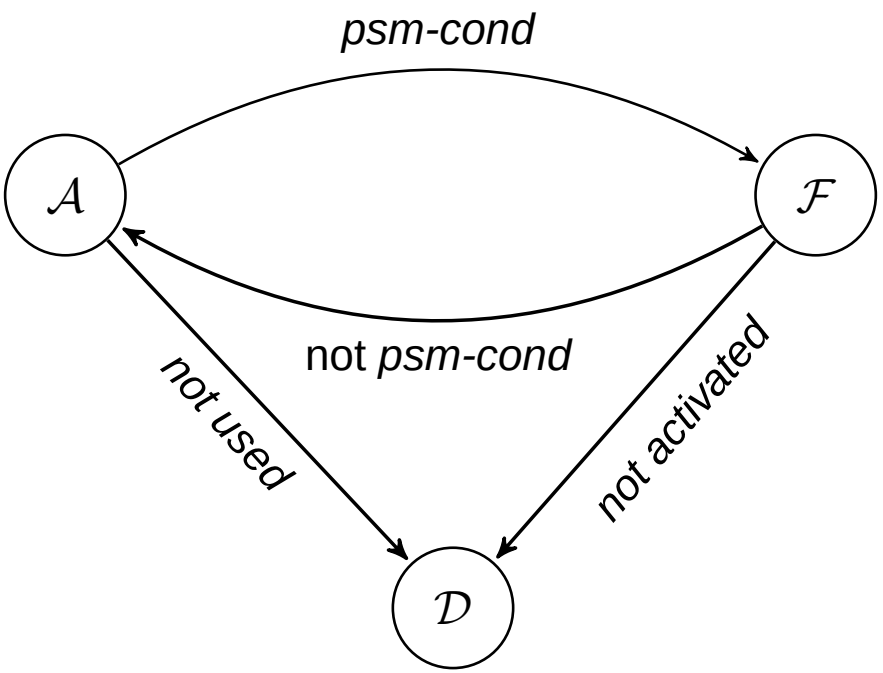
- $\mathcal{A}$: Set of active clauses (used in propagation)
- $\mathcal{D}$: Set of deleted clauses



Freezing learnt clauses

Freezing learnt clause management

- $\mathcal{A}$: Set of active clauses (used in propagation)
- $\mathcal{D}$: Set of deleted clauses
- $\mathcal{F}$: Set of frozen clauses



How can we solve SAT in parallel?

Two main approaches

How can we solve SAT in parallel?

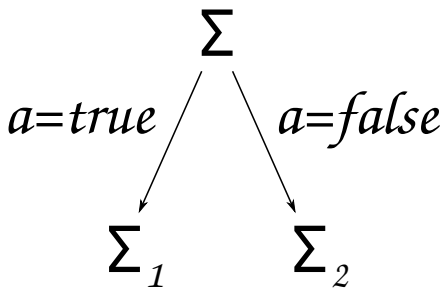
Two main approaches

Divide and conquer

How can we solve SAT in parallel?

Two main approaches

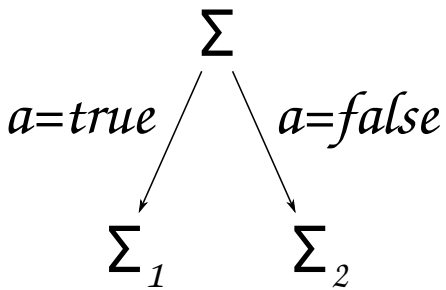
Divide and conquer



How can we solve SAT in parallel?

Two main approaches

Divide and conquer

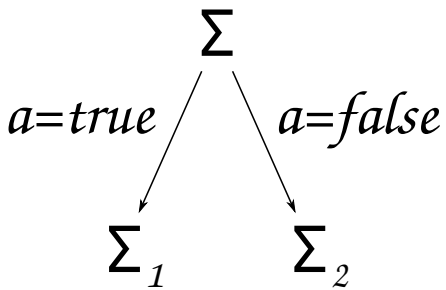


Portofolio

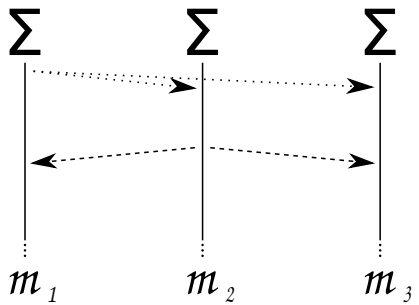
How can we solve SAT in parallel?

Two main approaches

Divide and conquer



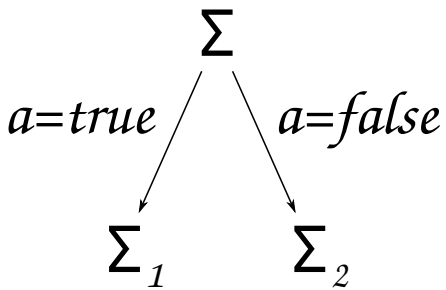
Portofolio



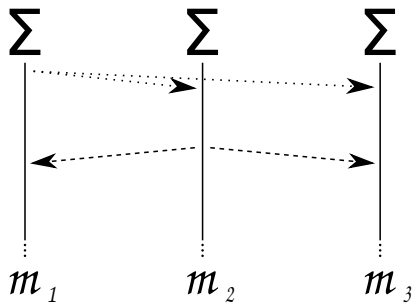
How can we solve SAT in parallel?

Two main approaches

Divide and conquer



Portofolio



We will use the portofolio methodology

SAT competition 2011

ppfolio

SAT competition 2011

ppfolio

- run completely different state-of-the-art solvers in parallel

SAT competition 2011

ppfolio

- run completely different state-of-the-art solvers in parallel
- trusted the competition (16 medals)

SAT competition 2011

ppfolio

- run completely different state-of-the-art solvers in parallel
- trusted the competition (16 medals)
- the solvers do not communicate!

SAT competition 2011

ppfolio

- run completely different state-of-the-art solvers in parallel
- trusted the competition (16 medals)
- the solvers do not communicate!

Work needs to be done on communication

Good communication?

Good communication?

Good communication

To achieve good communication, we need to **maximize** the exchange of **useful** information, and **minimize** the **useless** information.

Communication in portofolio

Good communication

Communication in portofolio

Good communication

- Information = learnt clauses

Communication in portofolio

Good communication

- Information = learnt clauses
- What is a useful clause?

Communication in portofolio

Good communication

- Information = learnt clauses
- What is a useful clause?
- A useless clause is never used in propagation

New ratio

New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread t

New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread t
- $used(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread t

New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread t
- $used(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread t
- $unused(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread t without being used

New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread t
- $used(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread t
- $unused(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread t without being used
- $\#\mathcal{I}_t - used(\mathcal{I}_t, t) - unused(\mathcal{I}_t, t)$ the number of clauses in the database that are neither used, nor deleted by thread t

New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread t
- $used(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread t
- $unused(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread t without being used
- $\#\mathcal{I}_t - used(\mathcal{I}_t, t) - unused(\mathcal{I}_t, t)$ the number of clauses in the database that are neither used, nor deleted by thread t

Usage ratio

$$\frac{\sum_{t=0}^{n} used(\mathcal{I}_t, t)}{\sum_{t=0}^{n} \#\mathcal{I}_t}$$

New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread t
- $used(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread t
- $unused(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread t without being used
- $\#\mathcal{I}_t - used(\mathcal{I}_t, t) - unused(\mathcal{I}_t, t)$ the number of clauses in the database that are neither used, nor deleted by thread t

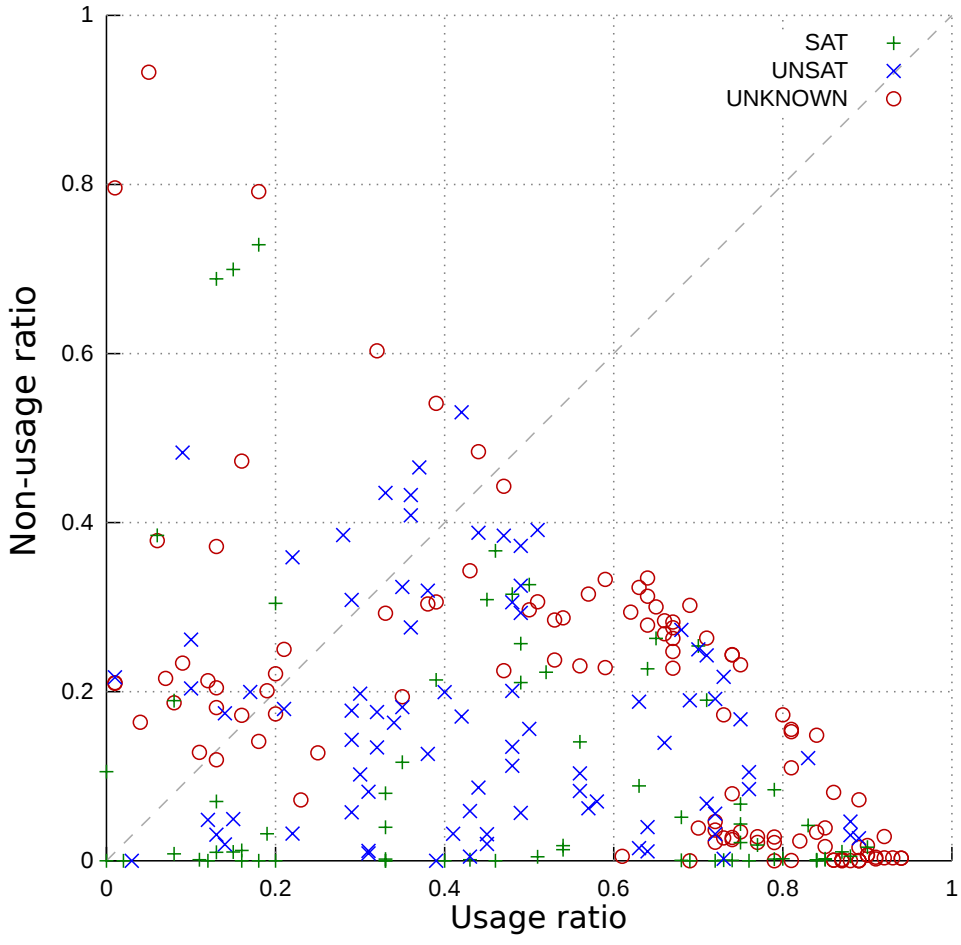
Usage ratio

$$\frac{\sum_{t=0}^{n} used(\mathcal{I}_t, t)}{\sum_{t=0}^{n} \#\mathcal{I}_t}$$

Non-usage ratio

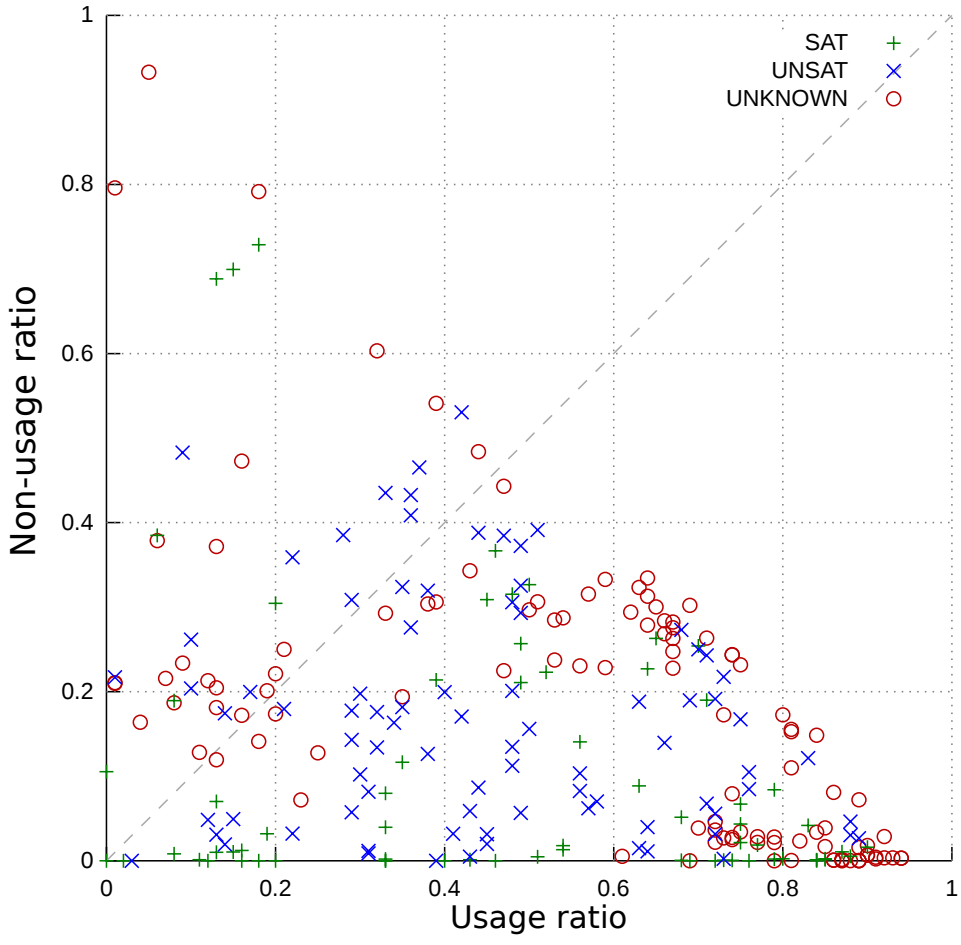
$$\frac{\sum_{t=0}^{n} unused(\mathcal{I}_t, t)}{\sum_{t=0}^{n} \#\mathcal{I}_t}$$

Classic manysat



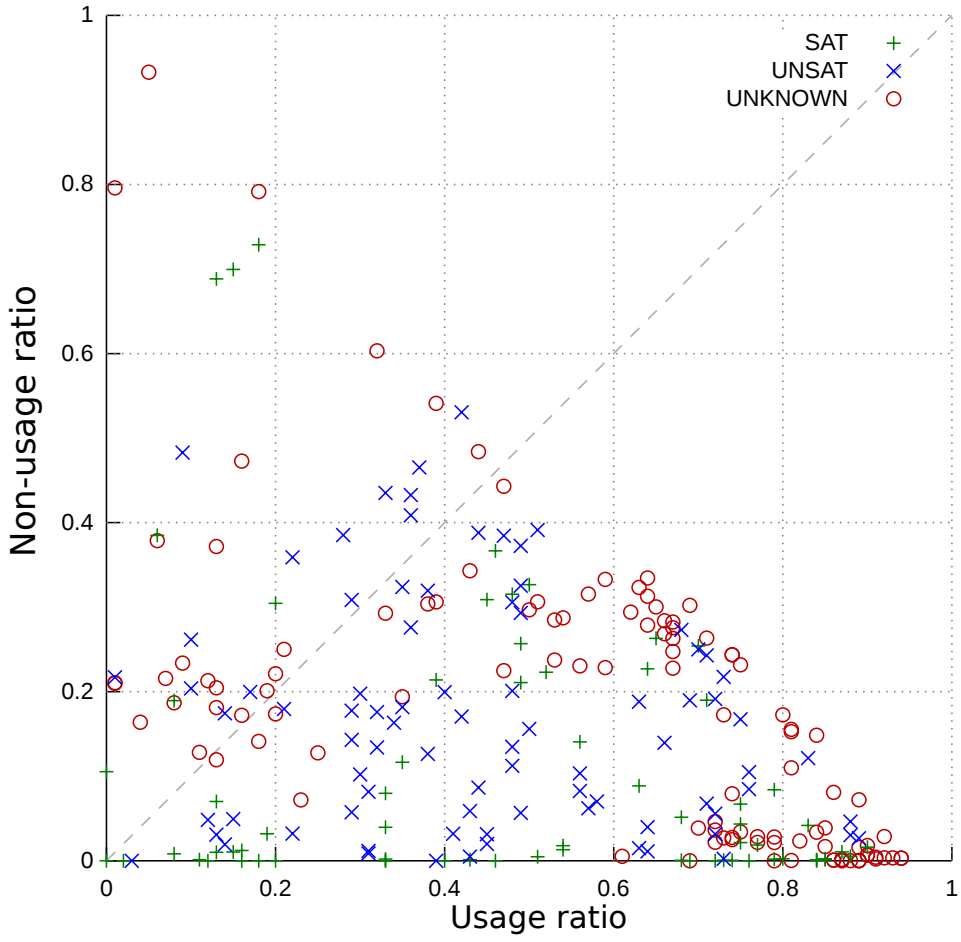
Classic manysat

- Ratio is good



Classic manysat

- Ratio is good
- A lot of imported clauses are not used but kept in memory



Challenges

Problems we must face

Challenges

Problems we must face

- Importation of duplicate information

Challenges

Problems we must face

- Importation of duplicate information
- Imported clauses can be useless for the current search subspace

Challenges

Problems we must face

- Importation of duplicate information
- Imported clauses can be useless for the current search subspace
- Higher number of learnt clauses

Introducing PeneLoPe

We want to design a solver based on ManySat2.0 able to:

Introducing PeneLoPe

We want to design a solver based on ManySat2.0 able to:

- handle all the learnt clauses

Introducing PeneLoPe

We want to design a solver based on ManySat2.0 able to:

- handle all the learnt clauses
- communicate efficiently

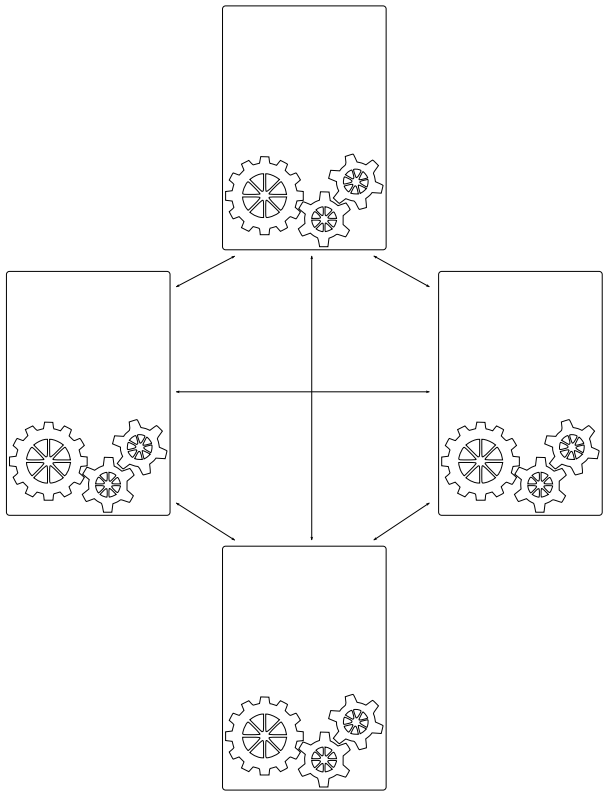
Introducing PeneLoPe

We want to design a solver based on ManySat2.0 able to:

- handle all the learnt clauses
- communicate efficiently
- use every processor on the host

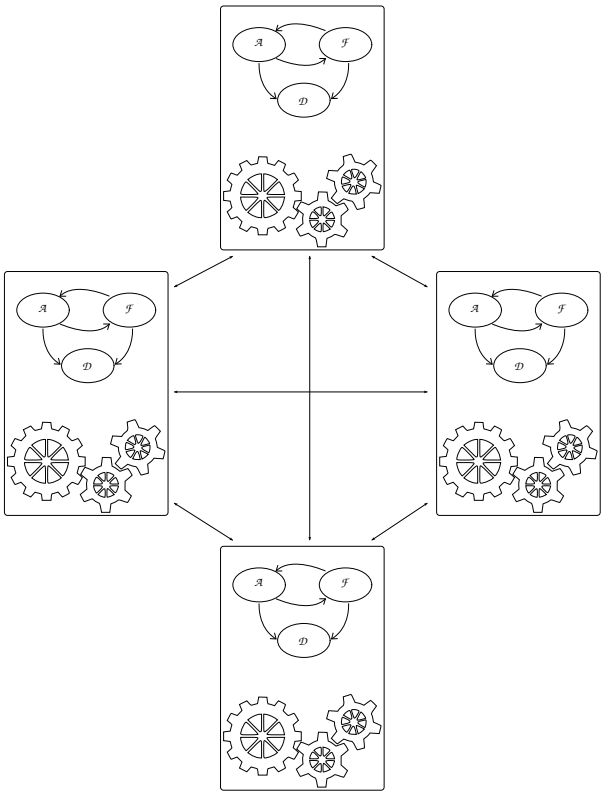
Freeze in parallel

Freeze in parallel



Freeze in parallel

- Each thread has its own sets



Import policy

Import policy

- Freeze-all

Import policy

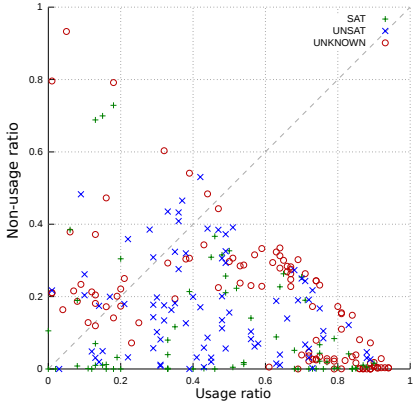
- Freeze-all
- Freeze

Import policy

- Freeze-all
- Freeze
- No freeze

Affecting the ratio

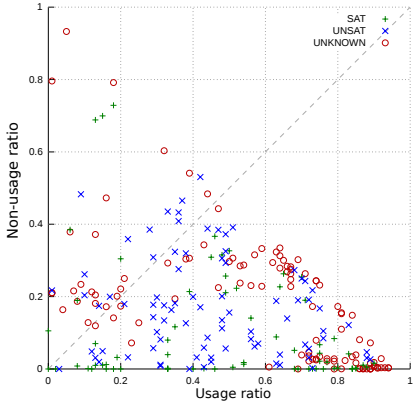
We could change the ratio by:



Affecting the ratio

We could change the ratio by:

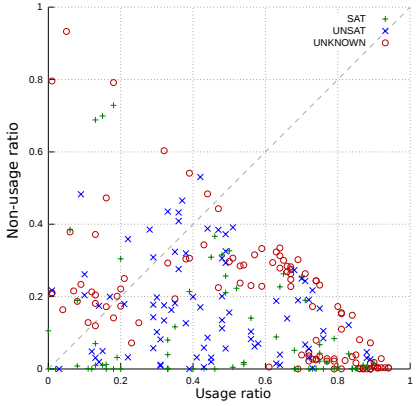
- Restart strategy



Affecting the ratio

We could change the ratio by:

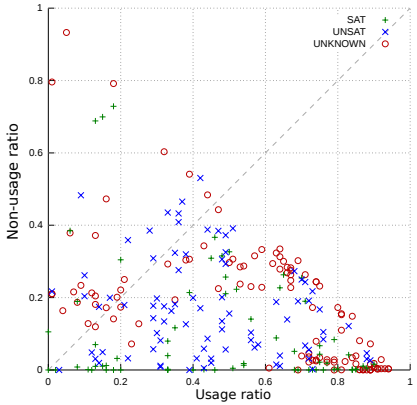
- Restart strategy
  -Luby technique



Affecting the ratio

We could change the ratio by:

- Restart strategy
  - -Luby technique
  - -*lbd* restarts (*glucose*)

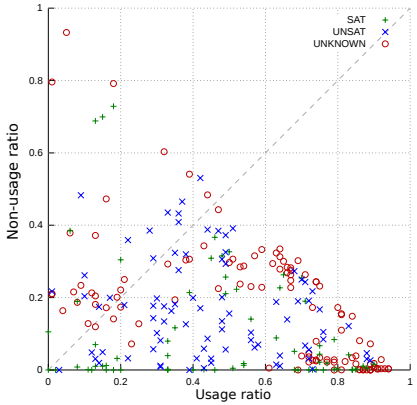


Affecting the ratio

We could change the ratio by:

- Restart strategy
  - -Luby technique
  - -*lbd* restarts (*glucose*)

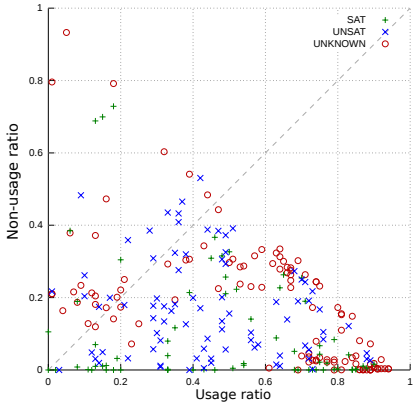
- Choosing what is exported



Affecting the ratio

We could change the ratio by:

- Restart strategy
  - -Luby technique
  - -*lbd* restarts (*glucose*)

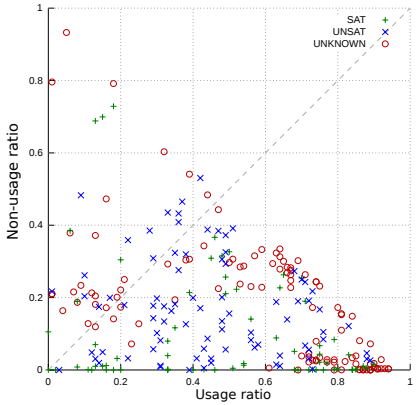


- Choosing what is exported
  - -Export every generated clauses

Affecting the ratio

We could change the ratio by:

- Restart strategy
  - -Luby technique
  - -*lbd* restarts (*glucose*)

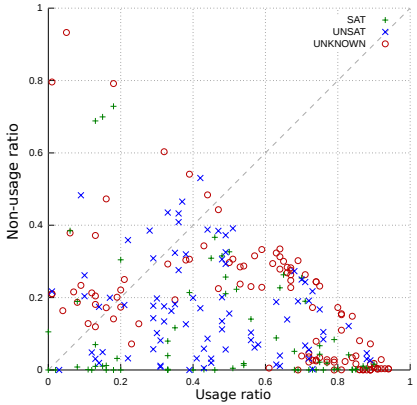


- Choosing what is exported
  - -Export every generated clauses
  - -Export clauses of size $\leq s$

Affecting the ratio

We could change the ratio by:

- Restart strategy
  - -Luby technique
  - -*lbd* restarts (*glucose*)



- Choosing what is exported
  - -Export every generated clauses
  - -Export clauses of size $\leq s$
  - -Export clauses with literal block distance $\leq l$

A closer look

SLN: size based export, luby restarts, no freeze at import
LLF: lbd based export, lbd restarts, freeze at import

instance	version	time	nb_i/nb_c	nb_i	nb_f	nb_u
hwmcc10-...	SLN	TO	1.34	7989	0%	35%
k50-eijkbs6669-tseitin	LLF	766	4.55	15299	10%	11%
AProVE07-21	SLN	10	1.06	83	0%	35%
	LLF	31	3.53	506	89%	9%

nb_c: number of conflicts (in thousands)
nb_i: number of imported learnt clauses (in thousands)
nb_f: the percentage of learnt clauses frozen at the import
nb_u: the percentage of used learnt clauses

A closer look

SLN: size based export, luby restarts, no freeze at import
LLF: lbd based export, lbd restarts, freeze at import

instance	version	time	nb_i/nb_c	nb_i	nb_f	nb_u
hwmcc10-...	SLN	TO	1.34	7989	0%	35%
k50-eijkbs6669-tseitin	LLF	766	4.55	15299	10%	11%
AProVE07-21	SLN	10	1.06	83	0%	35%
	LLF	31	3.53	506	89%	9%

nb_c: number of conflicts (in thousands)
nb_i: number of imported learnt clauses (in thousands)
nb_f: the percentage of learnt clauses frozen at the import
nb_u: the percentage of used learnt clauses

Comparison of combinations

Comparison of combinations

- policies have effects on each other

Comparison of combinations

- policies have effects on each other
- winning policy on our experiments:
  - export: *lbd* based
  - import: *no freeze*
  - restarts: *lbd* based.

Comparison of combinations

- policies have effects on each other
- winning policy on our experiments:
  - export: *lbd* based
  - import: *no freeze*
  - restarts: *lbd* based.

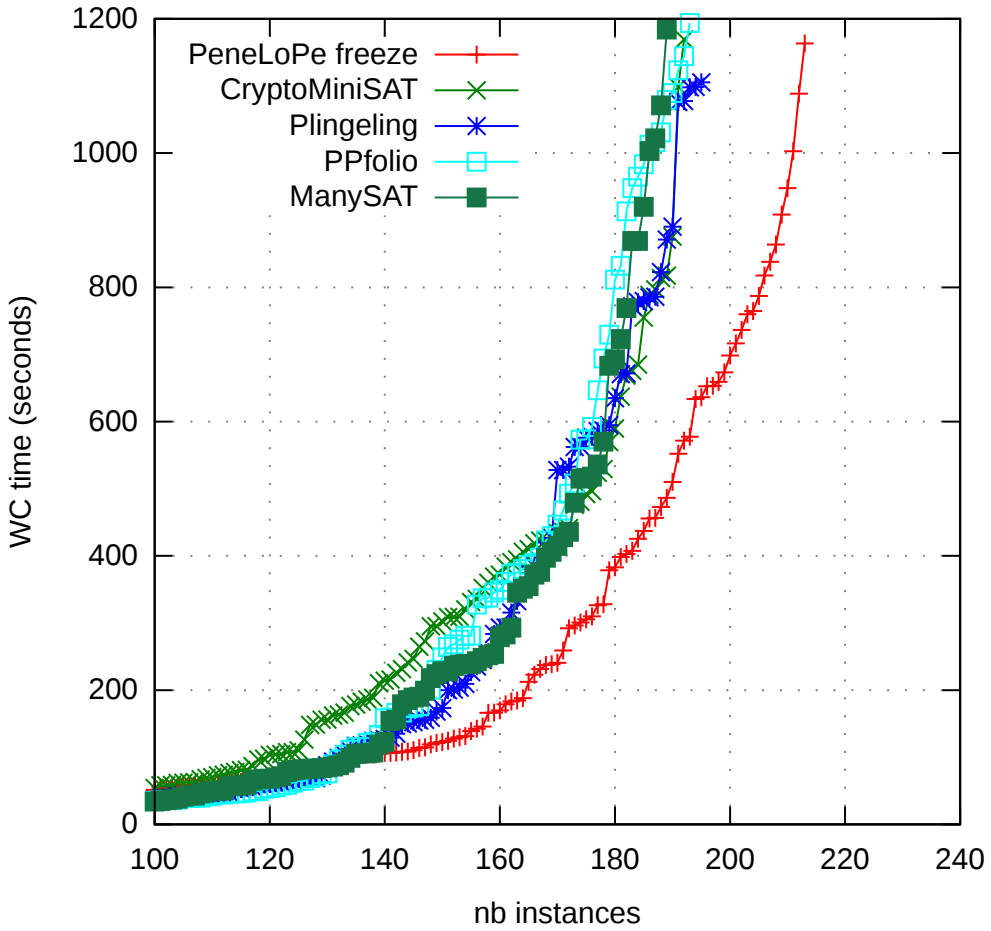
	#SAT	#UNSAT	#SAT + #UNSAT
Manysat	95	93	188
PeneLoPe			

Comparison of combinations

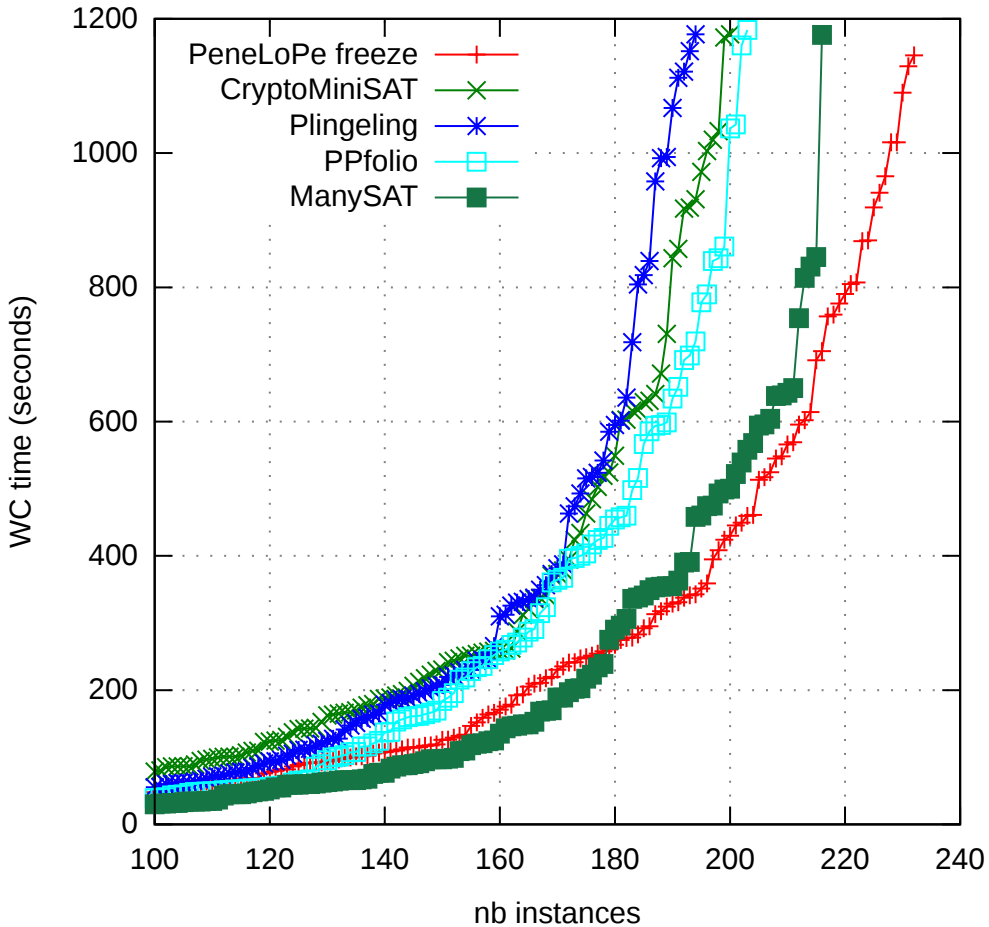
- policies have effects on each other
- winning policy on our experiments:
  - export: *lbd* based
  - import: *no freeze*
  - restarts: *lbd* based.

	#SAT	#UNSAT	#SAT + #UNSAT
Manysat	95	93	188
PeneLoPe	94	111	205

Comparison with other solvers (8 cores)



Scaling up to 32 cores



Conclusion

Conclusion

- We need to pay attention to clause exchange technique

Conclusion

- We need to pay attention to clause exchange technique
- The prototype is highly competitive

Conclusion

- We need to pay attention to clause exchange technique
- The prototype is highly competitive
- We can expend the orthogonality of the threads by using different techniques for each thread

Thank you for your attention

Questions?

Some comparisons

psm used	export strategy	restart strategy	import strategy	#SAT	#UNSAT	#SAT + #UNSAT
✓	*lbd limit*	*lbd*	*no freeze*	94	111	**205**
✓	*lbd limit*	*lbd*	*freeze*	89	**113**	202
✓	*size limit*	*lbd*	*freeze*	93	107	200
✓	*size limit*	*lbd*	*no freeze*	89	107	196
✓	*size limit*	*luby*	*no freeze*	**97**	98	195
✓	*lbd limit*	*lbd*	*freeze all*	89	102	191
✓	*size limit*	*luby*	*freeze all*	96	92	188
✓	*unlimited*	*lbd*	*freeze*	86	102	188
✓	*size limit*	*luby*	*freeze*	92	96	188
✓	*lbd limit*	*luby*	*freeze*	91	97	188
manysat	-	-	-	95	93	188
✓	*lbd limit*	*luby*	*no freeze*	90	94	184
✓	*unlimited*	*luby*	*freeze*	91	92	183
	size limit	*luby*	*no freeze*	92	90	182
✓	*unlimited*	*luby*	*no freeze*	89	88	177
✓	*size = 1*	*lbd*	*freeze*	89	88	177

8 cores details

Solver	#SAT	#UNSAT	#SAT+#UNSAT
PeneLoPe *freeze*	97	**119**	**216**
PeneLoPe *no freeze*	96	**119**	215
plingeling	**99**	97	196
ppfolio	91	103	194
cryptominisat	89	104	193
ManySat	95	92	187

32 cores details

Solver	#SAT	#UNSAT	#SAT+#UNSAT
PeneLoPe *freeze*	104	127	**231**
PeneLoPe *no freeze*	99	**131**	230
ManySat	105	111	216
ppfolio	**107**	97	204
cryptominisat	96	105	201
plingeling	100	95	195

Literal block distance

Literal block distance

Definition

Given a clause $\mathcal{C}$, and a partition of its literals into n subsets according to the current assignment, s.t. literals are partitioned w.r.t their decision level. The *lbd* of $\mathcal{C}$ is exactly n.

Literal block distance

Definition

Given a clause $\mathcal{C}$, and a partition of its literals into n subsets according to the current assignment, s.t. literals are partitioned w.r.t their decision level. The *lbd* of $\mathcal{C}$ is exactly n.

lbd restarts

Avg_s is the average of *lbd* of the clauses created since the start of the process. Avg_{100} is the average of *lbd* over the last 100 created clauses. Restarts when $Avg_{100} \times \alpha \geq Avg_s$, $\alpha = 0.7$