Enhancing State-of-the-Art Parallel SAT Solvers Through Optimized Sharing Policies

Vincent Vallade[1] Julien Sopena[1] Souheib Baarir[2]

Sorbonne Université, CNRS, LIP6, UMR 7606, Paris, France[1]

EPITA, Toulouse, France[2]

Pragmatics of SAT 2023

Sequential enumeration reaches its limits

CDCL solvers are efficients thanks to:

- Preprocessing **[EB05, PHS08]**
- Branching Heuristics **[ZMMM01, LGPC16]**
- Qualifying learned clauses for garbage collection

Sequential enumeration reaches its limits

CDCL solvers are efficients thanks to:

- Preprocessing [EB05, PHS08]
- Branching Heuristics [ZMMM01, LGPC16]
- Qualifying learned clauses for garbage collection

Sequential solving reaches limits:

- Instances became bigger and more complex over time
- Rarity of new heuristics
- Hardware limits (end of Moore’s law)

Sequential enumeration reaches its limits

CDCL solvers are efficients thanks to:

- Preprocessing **[EB05, PHS08]**
- Branching Heuristics **[ZMMM01, LGPC16]**
- Qualifying learned clauses for garbage collection

Sequential solving reaches limits:

- Instances became bigger and more complex over time
- Rarity of new heuristics
- Hardware limits (end of Moore's law)

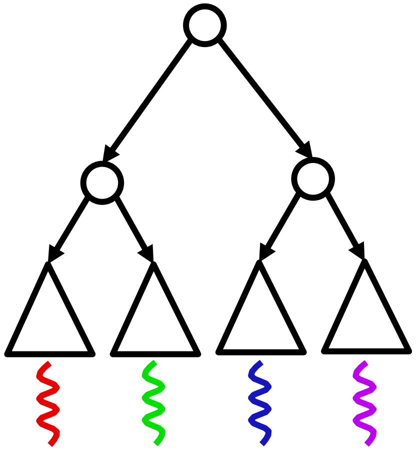
Developping parallel SAT solvers able to exploit new multicore machines become a necessity.

Interest in parallel SAT solving

Interest in parallel SAT solving

Divide and conquer

[ZBH96]

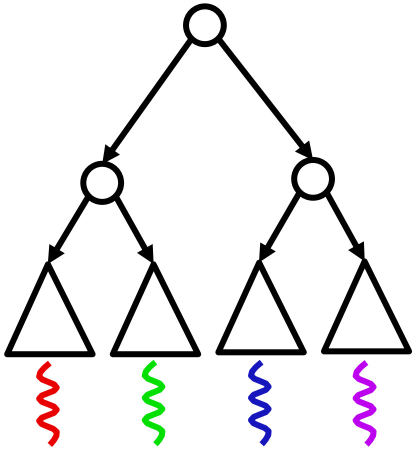


Dynamically divide the search space between each worker

Interest in parallel SAT solving

Divide and conquer

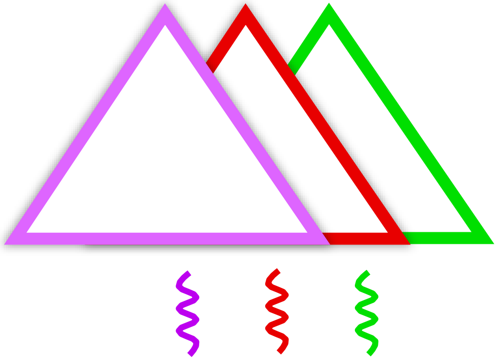
[ZBH96]



Dynamically divide the search space between each worker

Portfolio

[HJS09]

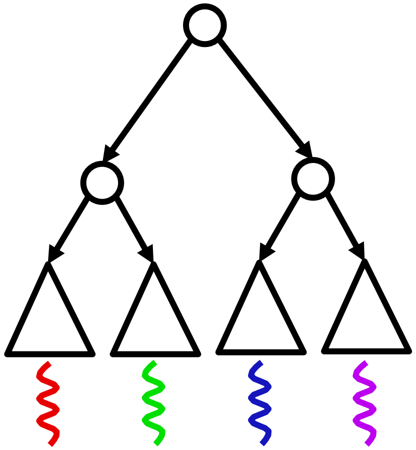


Multiple workers on the whole search space

Interest in parallel SAT solving

Divide and conquer

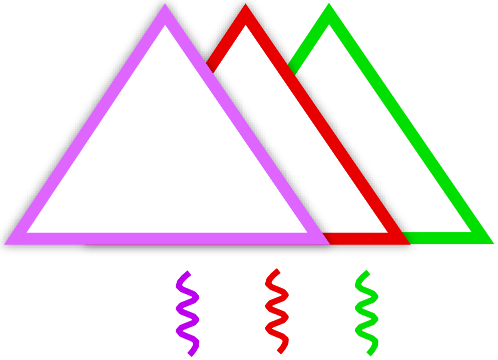
[ZBH96]



Dynamically divide the search space between each worker

Portfolio

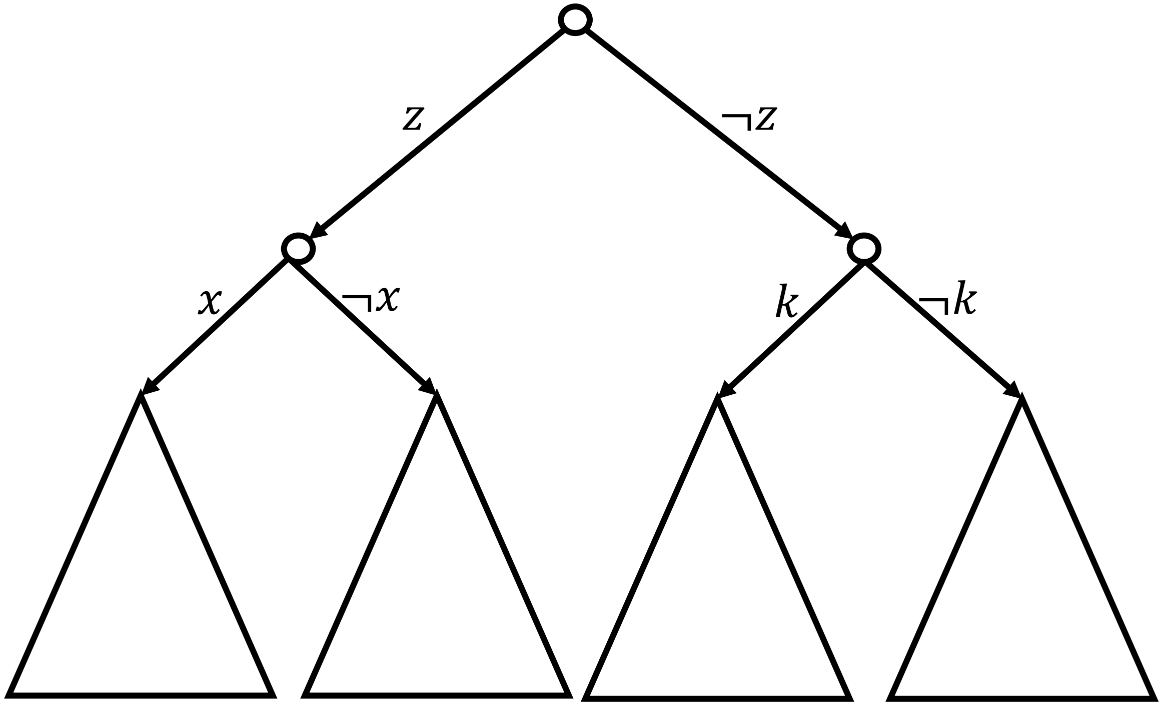
[HJS09]



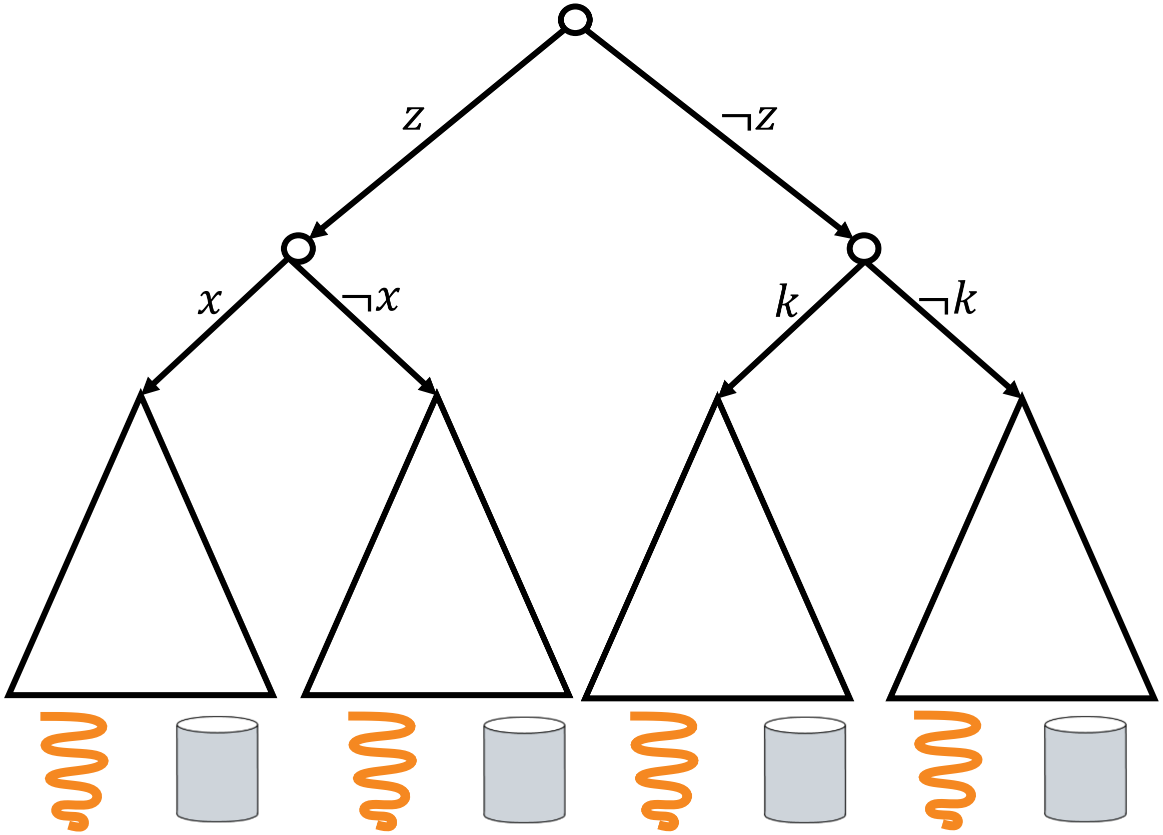
Multiple workers on the whole search space

Information can be shared between the solvers.

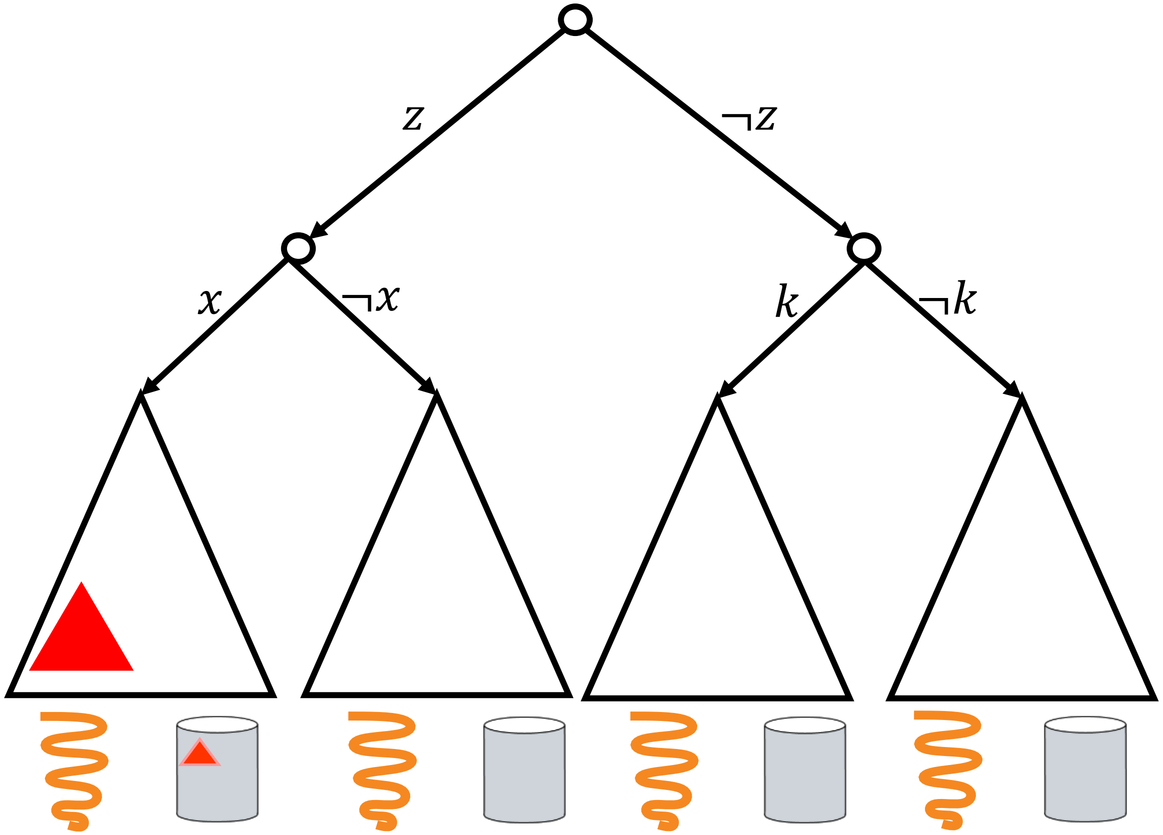
Clause Sharing



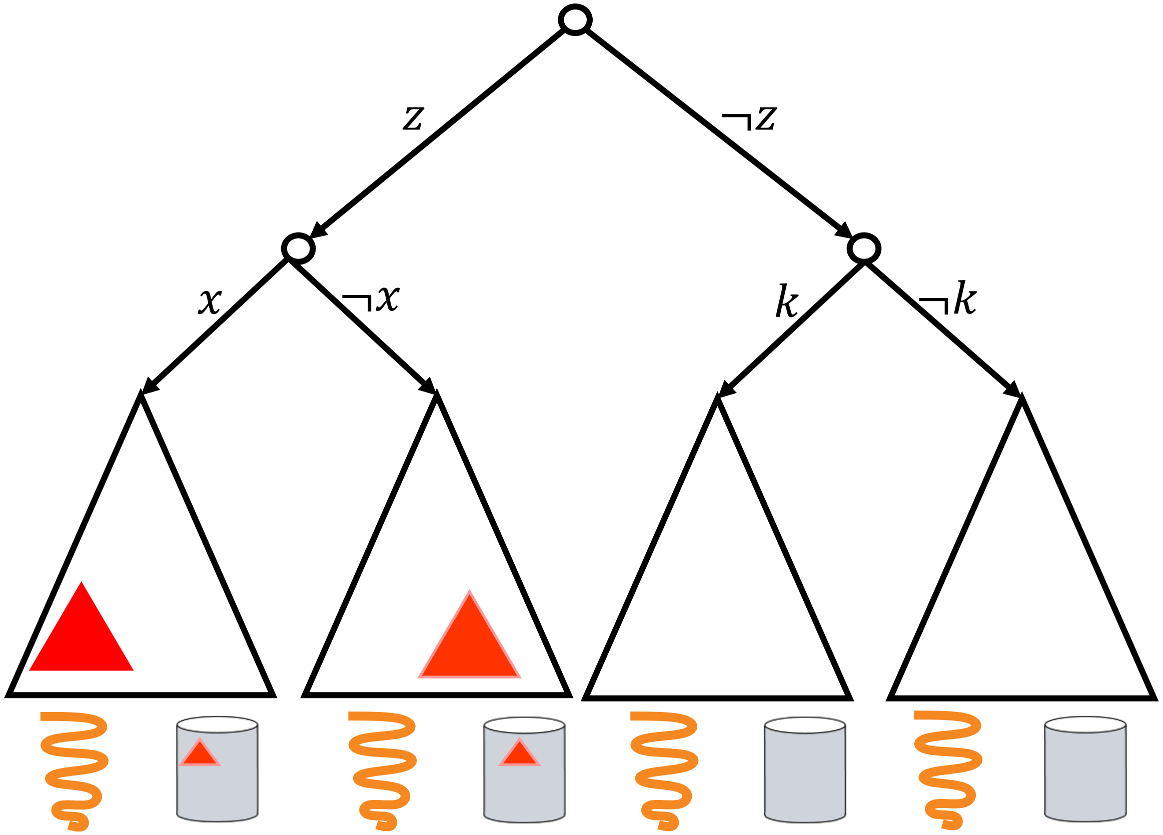
Clause Sharing



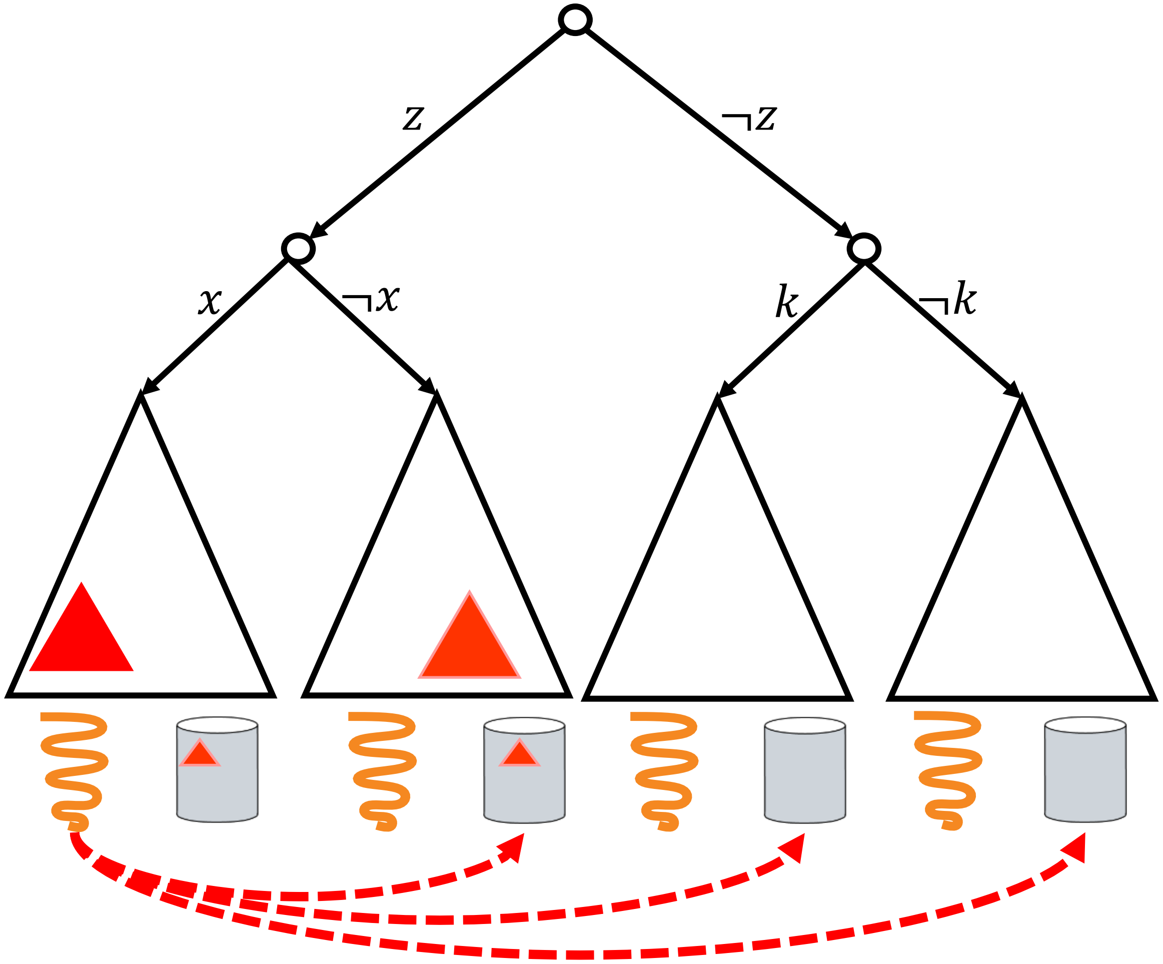
Clause Sharing



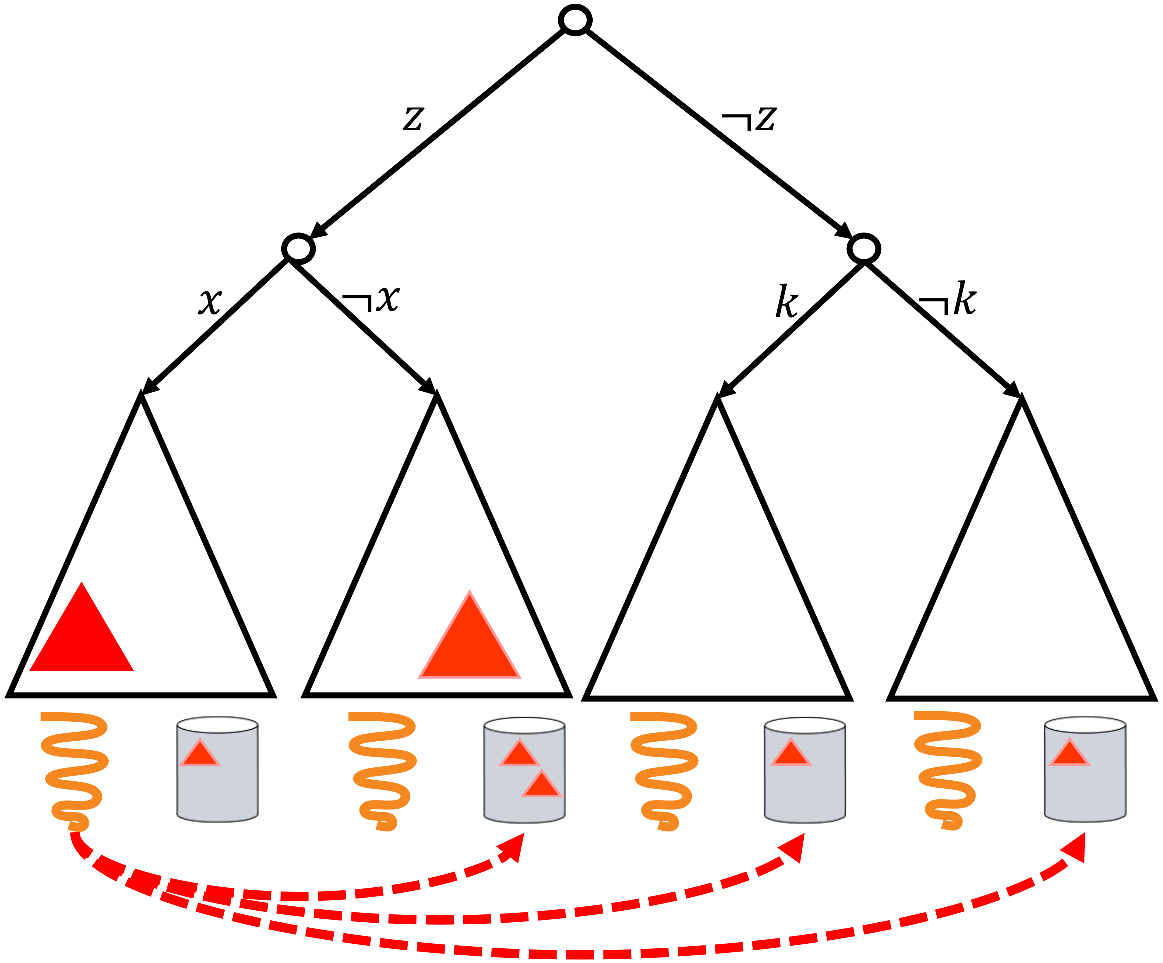
Clause Sharing



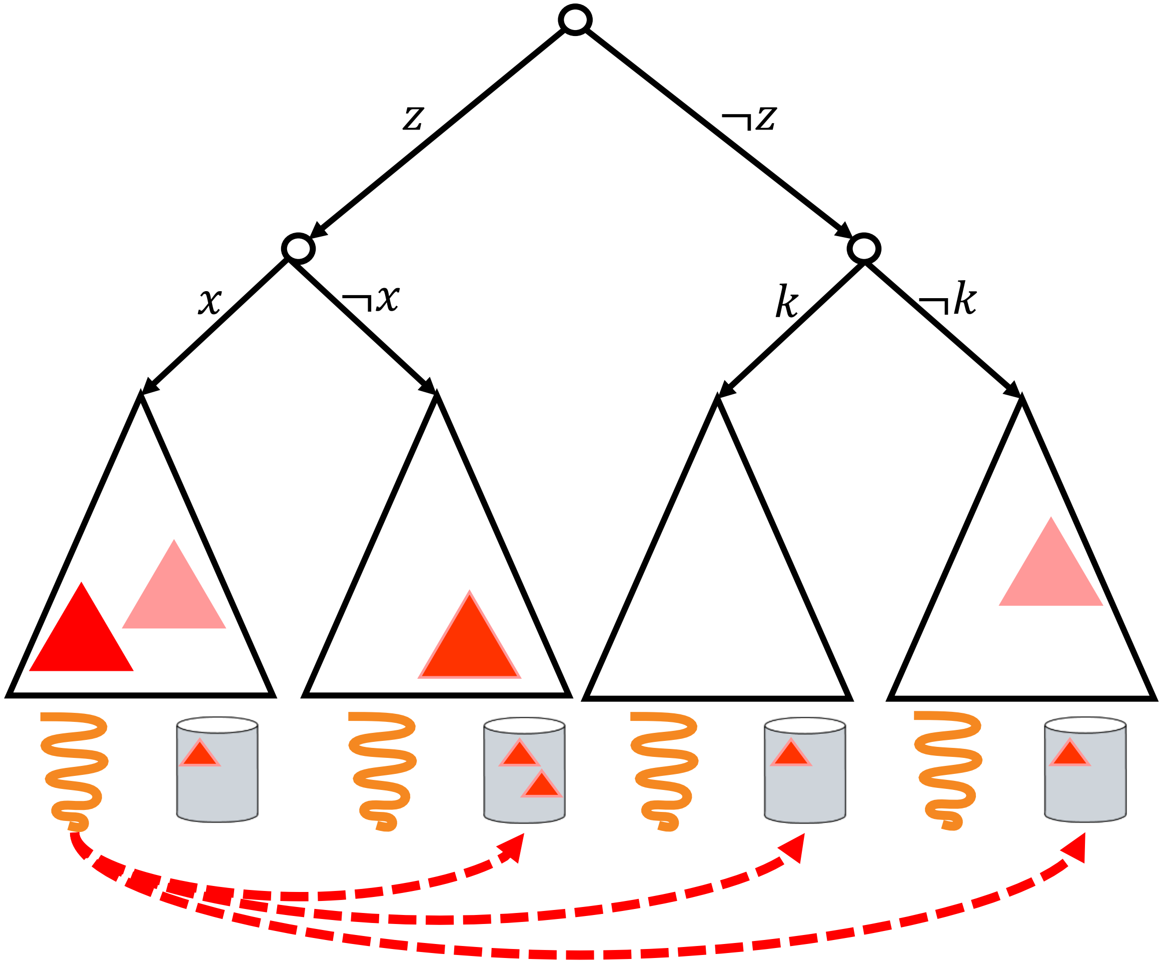
Clause Sharing



Clause Sharing



Clause Sharing



Challenges

In practice, sharing too much clauses can impair performances.

Algorithmic reasons:

- slow down unit propagation
- slow down garbage collection

Concurrency/Hardware reasons:

- memory contention (cacheline, alloc)
- memory footprint
- synchronization

How to select clauses to find the right trade-offs between cost and gain ?
(One of the 7 challenges of parallel SAT solving [HW13])

Sharing in P-MCOMSPS

What is shared ?

Clauses with a low LBD value [AS09].

Sharing in P-MCOMSPS

What is shared ?
Clauses with a low LBD value [AS09].

To whom ?
All solvers in the portfolio are producers and consumers

Sharing in P-MCOMSPS

What is shared ?

Clauses with a low LBD value [AS09].

To whom ?

All solvers in the portfolio are producers and consumers

How ?

A thread applies the hordesat sharing strategy [BSS15]:

- **Try to share 1500 literals**
- **Prioritize small clauses**
- **Increase/decrease LBD threshold when producer under/overproduces**

Sharing in P-MCOMSPS

What is shared ?

Clauses with a low LBD value [AS09].

To whom ?

All solvers in the portfolio are producers and consumers

How ?

A thread applies the hordesat sharing strategy [BSS15]:

- **Try to share 1500 literals**
- **Prioritize small clauses**
- **Increase/decrease LBD threshold when producer under/overproduces**

Added mechanisms:

- **Bloom filter for exchanged learned clauses [SS21]**
- **Asynchronous clauses minimization [VFBSK20]**

Progressive integration of sharing strategies

P-MCOMSPS sequential engine does not use recent discoveries

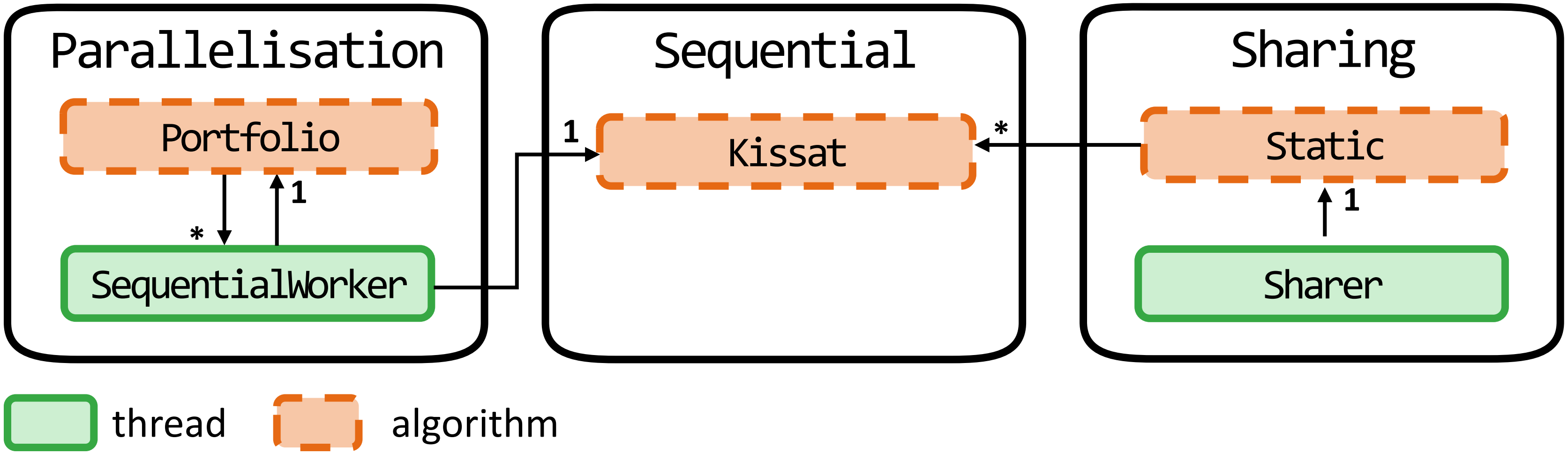
Kissat [BFFH20] dominates sequential and parallel solving

Contributions:

- **Implementation on another parallel SAT solver:**
  - We incrementally integrated each mechanism
  - We evaluated them on the SAT 2022 competition bench
  - We demonstrated combinations of mechanisms that improved performance
- **Scaling study performed on 48 and 64 core machines:**
  - Shows good scaling for SAT instances
  - Limited results for UNSAT instances

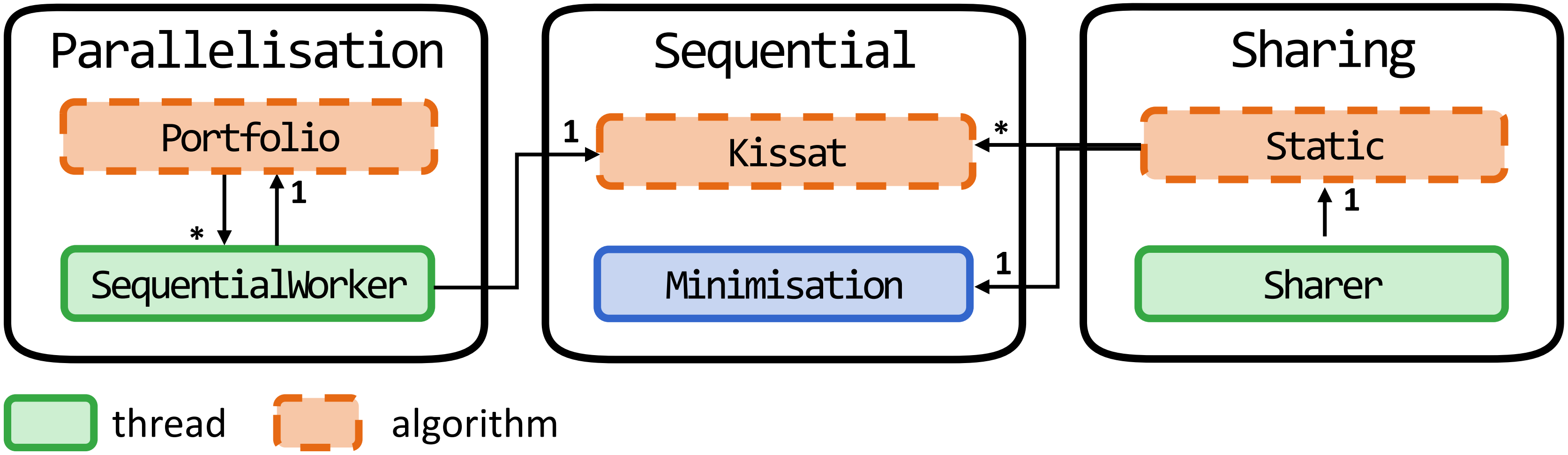
Incremental integration of sharing components

Parkissat: winner of the parallel track 2022 [ZCC22]



Incremental integration of sharing components

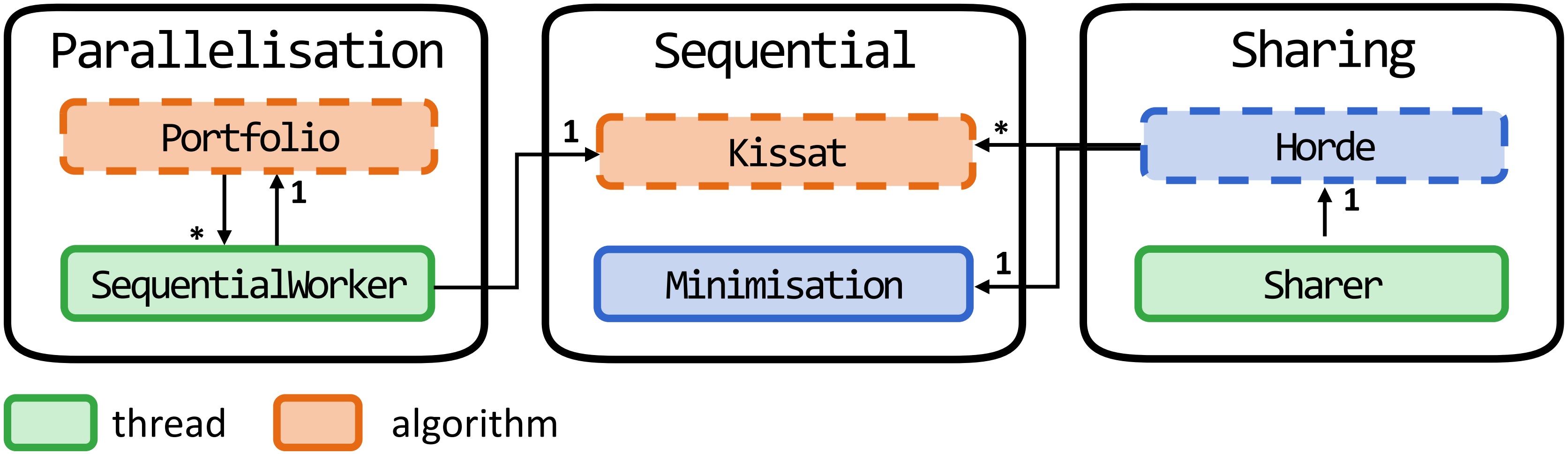
Parkissat: winner of the parallel track 2022 [ZCC22]



- **STR:** Asynchronous minimisation of clauses

Incremental integration of sharing components

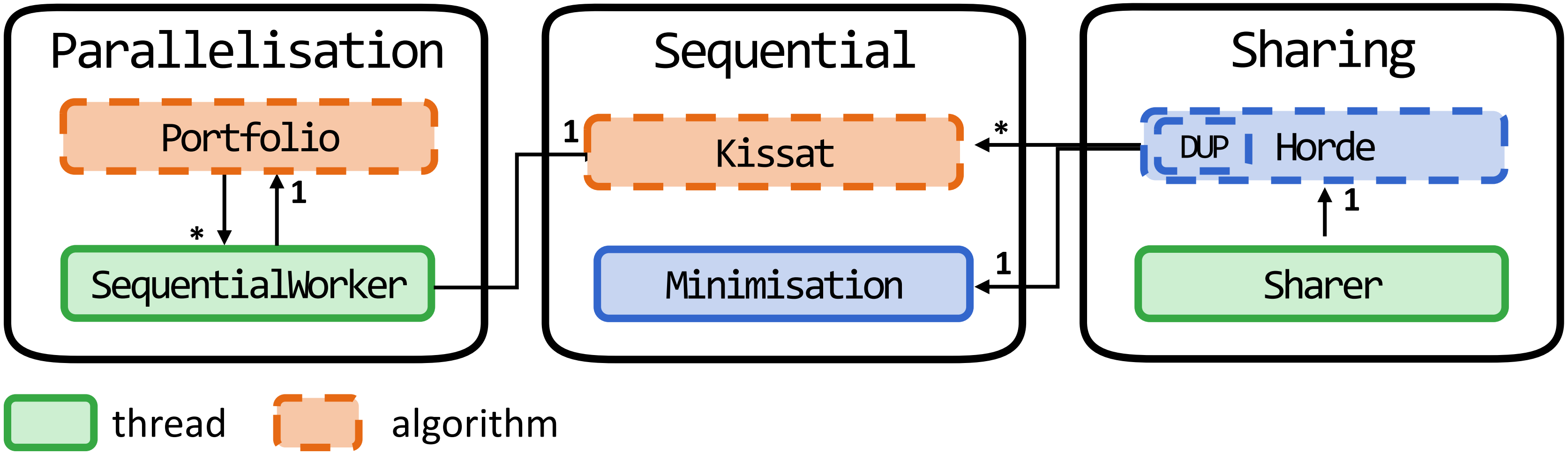
Parkissat: winner of the parallel track 2022 [ZCC22]



- **STR:** Asynchronous minimisation of clauses
- **Dynamic sharing strategy**: same than P-MCOMSPS

Incremental integration of sharing components

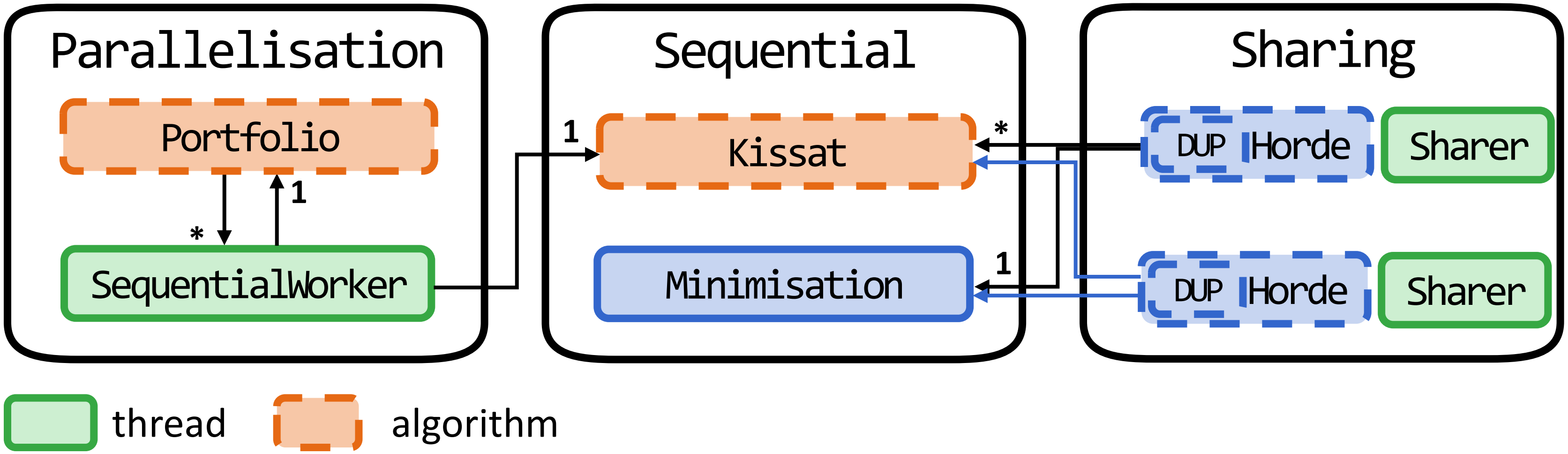
Parkissat: winner of the parallel track 2022 [ZCC22]



- **STR:** Asynchronous minimisation of clauses
- **Dynamic sharing strategy:** same than P-MCOMSPS
- **Dup:** Bloom filter for exchanged clauses

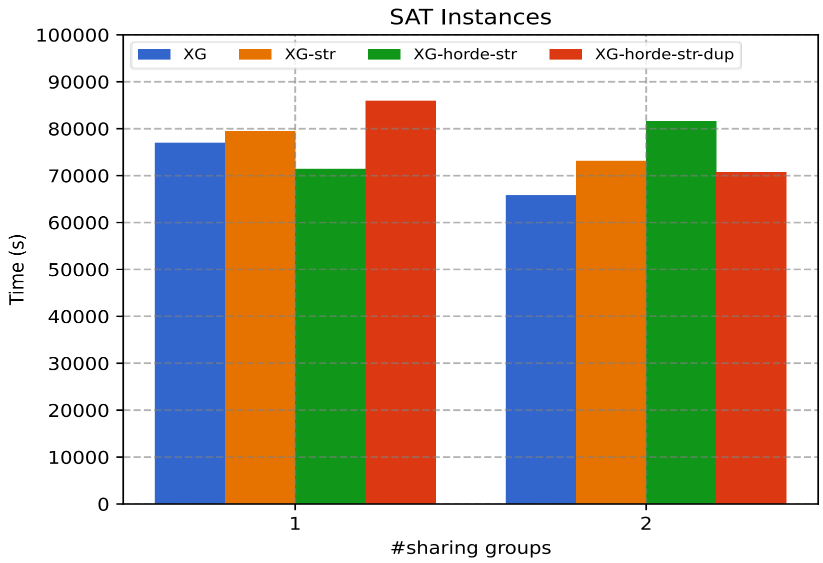
Incremental integration of sharing components

Parkissat: winner of the parallel track 2022 [ZCC22]



- **STR:** Asynchronous minimisation of clauses
- **Dynamic sharing strategy:** same than P-MCOMSPS
- **Dup:** Bloom filter for exchanged clauses
- **2G:** Add of a second sharing thread

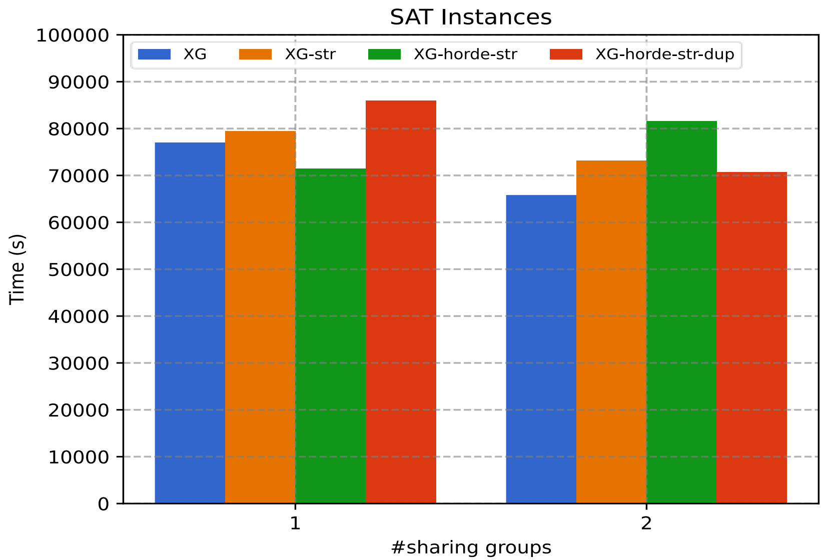
Performance study

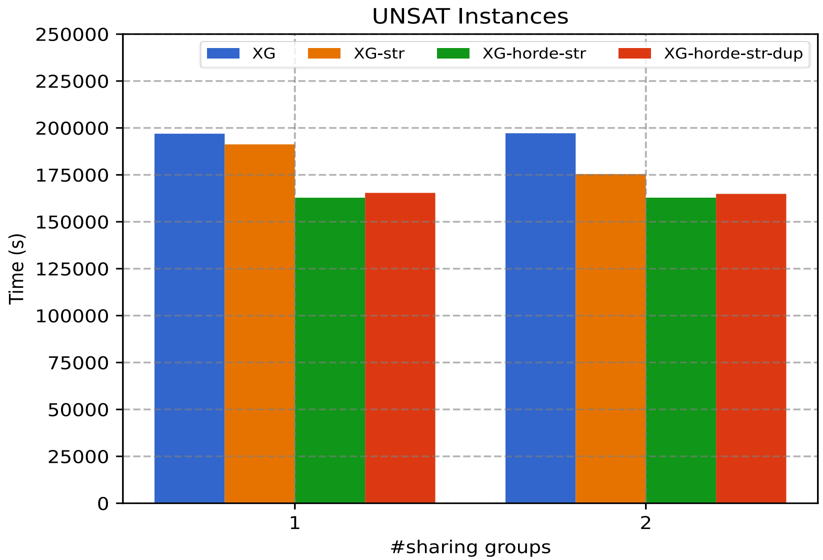


SAT instances: race to the solution

- Using more threads for sharing is useful
- Other heuristics are not fruitful

Performance study





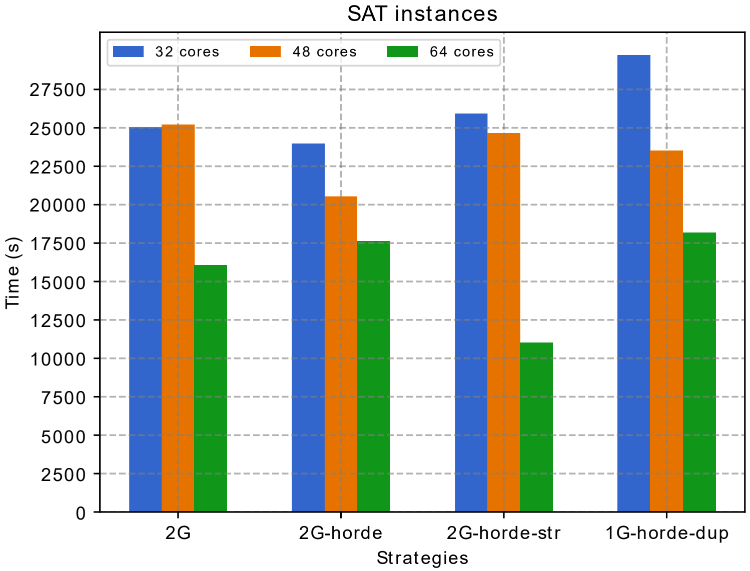
SAT instances: race to the solution

- Using more threads for sharing is useful
- Other heuristics are not fruitful

UNSAT instances: race to the unsat core

- Strengthening and horde improve performance
- Duplicates management does not

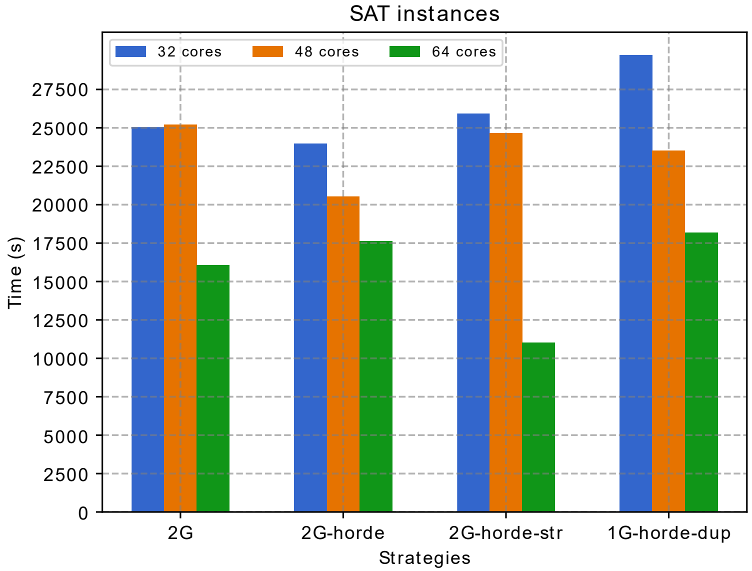
Scaling study

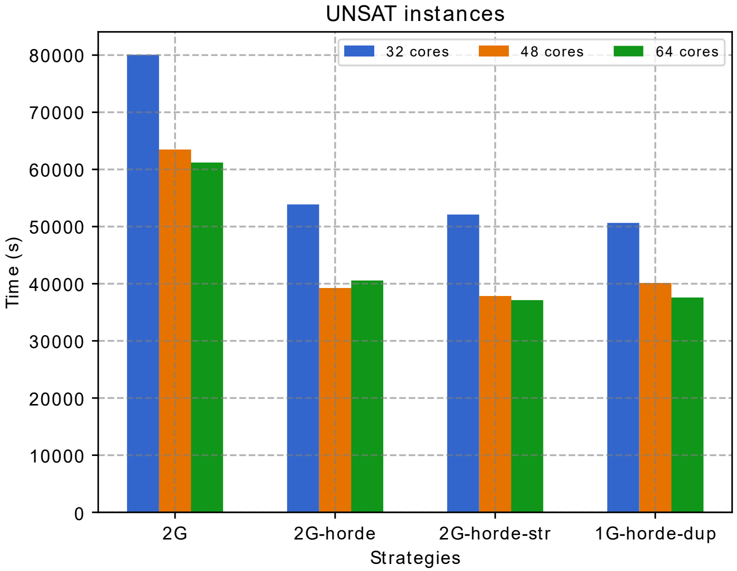


SAT instances:

The probability of finding a solution increases with the number of threads.

Scaling study





SAT instances:

The probability of finding a solution increases with the number of threads.

UNSAT instances:

Each path must be explored, more computational resources do not remove algorithm limitations.

Conclusion

Contributions :

- Evaluation of multiple sharing strategies
- Boost the performance of the best parallel solver
- Detect a scaling problem in the UNSAT resolution for this solver

Bibliography

- Niklas Eén and Armin Biere.
  Effective preprocessing in sat through variable and clause elimination.
  In Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT), pages 61–75. Springer, 2005.

- Cédric Piette, Youssef Hamadi, and Lakhdar Saïs.
  Vivifying propositional clausal formulae.
  In Proceedings of the 2008 Conference on ECAI 2008 : 18th European Conference on Artificial Intelligence, page 525–529, NLD, 2008. IOS Press.

- Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.
  Chaff Engineering an efficient sat solver.
  In Proceedings of the 38th Design Automation Conference (DAC), pages 530–535. ACM, 2001.

- Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki.
  Learning rate based branching heuristic for sat solvers.
  In Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT), pages 123–140. Springer, 2016.

Bibliography

- Lintao Zhang, Conor F Madigan, Matthew H Moskewicz, and Sharad Malik.
  Efficient conflict driven learning in a boolean satisfiability solver. *In Proceedings of the 20thIEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 279–285. IEEE, 2001.*

- Youssef Hamadi, Said Jabbour, and Lakhdar Sais.
  Manysat : a parallel sat solver.
  Journal on Satisfiability, Boolean Modeling and Computation, pages 245–262, 2009.

- Youssef Hamadi and Christoph Wintersteiger.
  Seven challenges in parallel sat solving.
  AI Magazine, 34(2) :99, Jun. 2013.

- Gilles Audemard and Laurent Simon.
  Predicting learnt clauses quality in modern sat solvers.
  In Proceedings of the 21st International Joint Conferences on Artifical Intelligence(IJCAI), pages 399–404. AAAI Press, 2009.

Bibliography

- Tomáš Balyo, Peter Sanders, and Carsten Sinz.
  Hordesat : A massively parallel port-folio sat solver.
  In Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing (SAT), pages 156–172. Springer, 2015.

- Dominik Schreiber and Peter Sanders.
  Scalable sat solving in the cloud.
  In Chu-Min Liand Felip Manyà, editors, Theory and Applications of Satisfiability Testing – SAT 2021,pages 518–534, Cham, 2021. Springer International Publishing.

- Vincent Vallade, Ludovic Le Frioux, Souheib Baarir, Julien Sopena, and Fabrice Kordon.
  On the usefulness of clause strengthening in parallel sat solving.
  In Ritchie Lee, Susmit Jha, Anastasia Mavridou, and Dimitra Giannakopoulou, editors, NASA Formal Methods, pages 222–229, Cham, 2020. Springer International Publishing.

- *Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger.*
  Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020.
  In Proceedings of sat competition 2020 : Solver and benchmark descriptions. University of Helsinki, Department of Computer Science, 2020

Bibliography

- Xindi Zhang, Zhihan Chen, and Shaowei Cai.
  Parkissat : Random shuffle based and preprocessing extended parallel solvers with clause sharing.
  In Proceedings of SAT Competition 2022 : Solver and Benchmark Descriptions, page 51. Department of Computer Science, University of Helsinki, Finland, 2022.