Modern Cooperative Parallel SAT Solving

Ahmed Irfan Davide Lanti Norbert Manthey

Technische Universität Dresden, Germany

July 8, 2013

Motivation

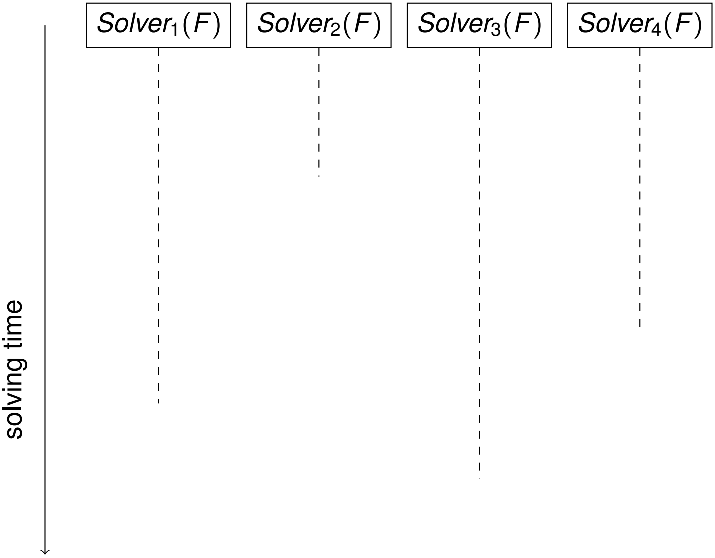
- Parallel architecture
- Over the last few year, less research on search space partitioning solvers

Parallel SAT Solving

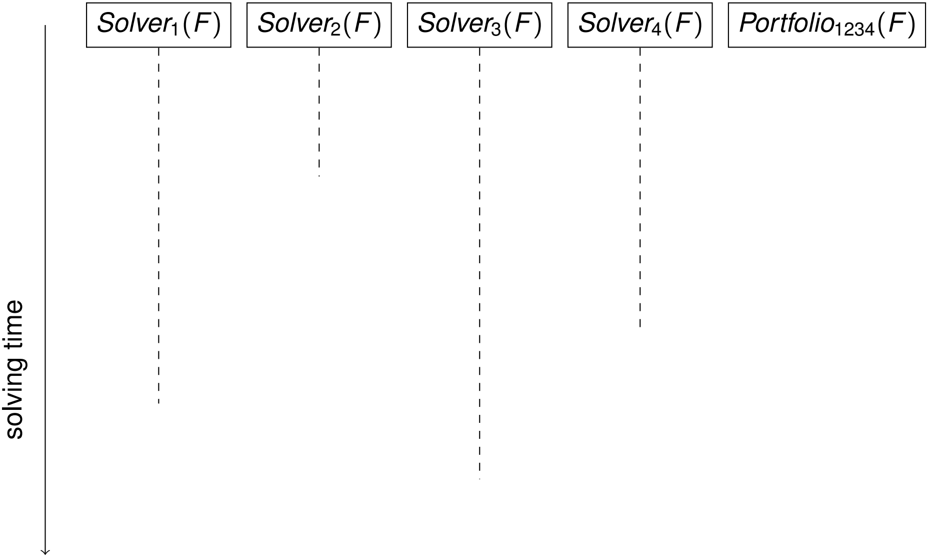
Two major techniques

- Portfolio - *(Competitive)*
- Search Space Partitioning - *(Cooperative)*

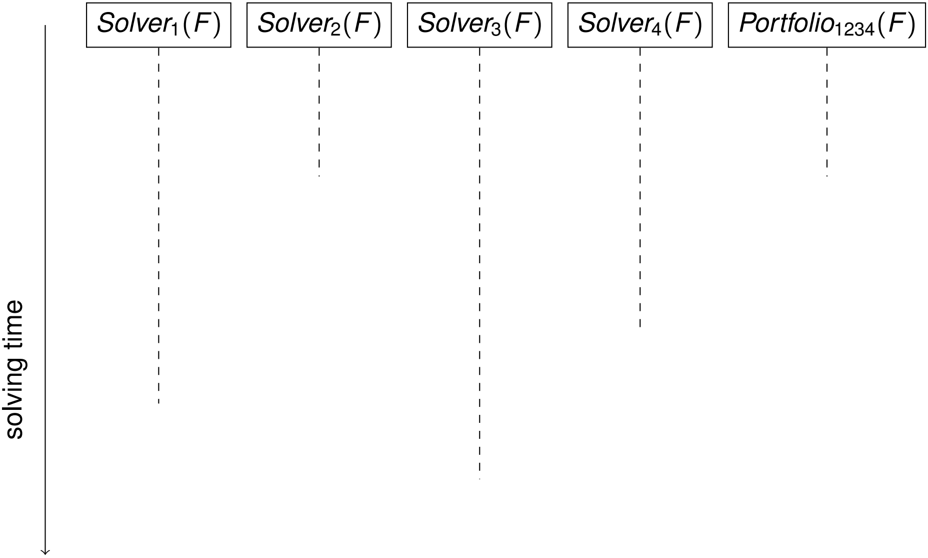
Portfolio Solver



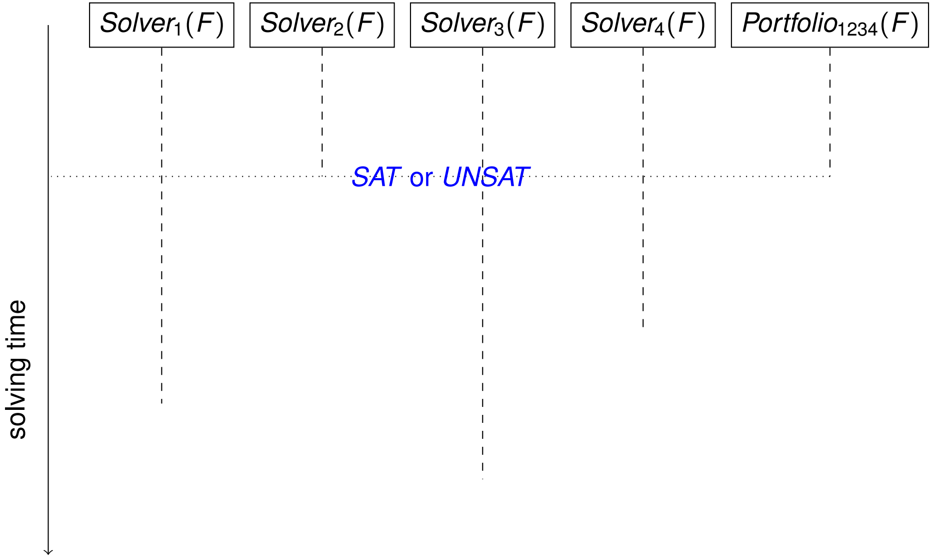
Portfolio Solver



Portfolio Solver



Portfolio Solver



Search Space Partitioning Solver

- Creating Partitions

- Solving Partitions

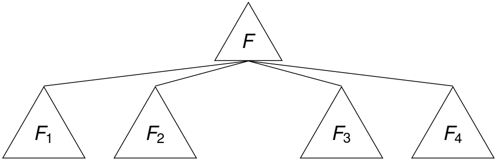
Search Space Partitioning Solver

- Creating Partitions
  Partition Function: $pf(F, n) = \{F_1, F_2, \ldots, F_n\}$
  such that $F \equiv F_1 \vee F_2 \vee \ldots \vee F_n$
  - F is satisfiable iff there exists $F' \in pf(F, n)$ is satisfiable
  - F is unsatisfiable iff all $F' \in pf(F, n)$ are unsatisfiable
  - Partitions are disjoint, i.e. for $1 \leq i < j \leq n$, $F_i \wedge F_j$ is unsatisfiable
- Solving Partitions

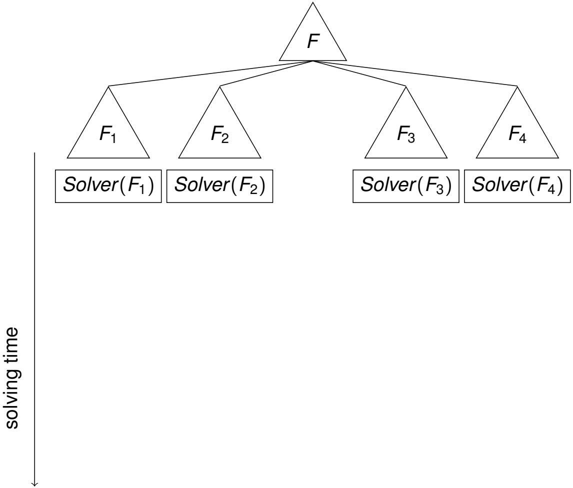
Search Space Partitioning Solver

- Creating Partitions
  Partition Function: $pf(F, n) = \{F_1, F_2, \ldots, F_n\}$
  such that $F \equiv F_1 \vee F_2 \vee \ldots \vee F_n$
  - F is satisfiable iff there exists $F' \in pf(F, n)$ is satisfiable
  - F is unsatisfiable iff all $F' \in pf(F, n)$ are unsatisfiable
  - Partitions are disjoint, i.e. for $1 \leq i < j \leq n$, $F_i \wedge F_j$ is unsatisfiable
- Solving Partitions
  - Non-overlapped solving (plain partitioning)
  - Overlapped solving (iterative partitioning)

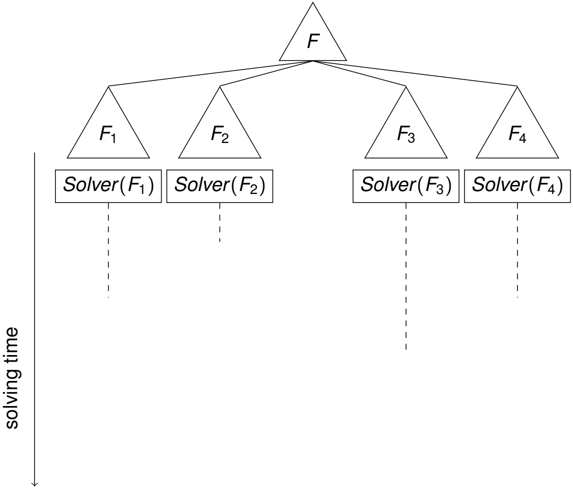
Search Space Partitioning Solver



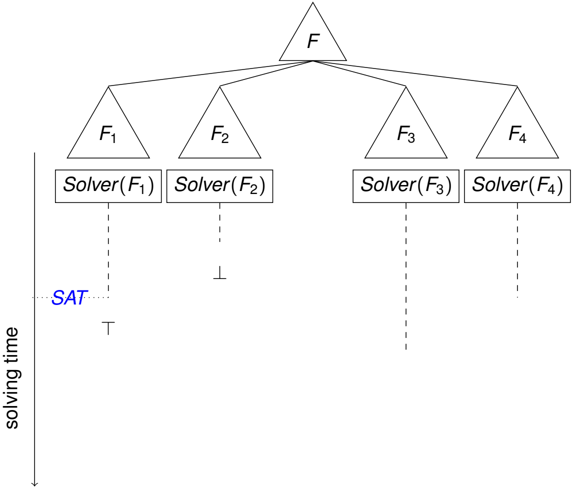
Search Space Partitioning Solver



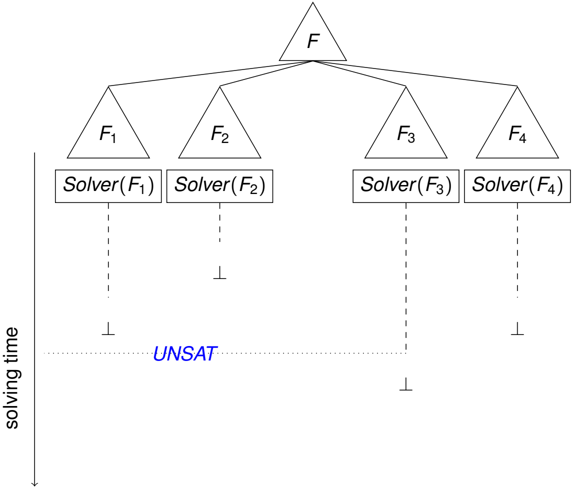
Search Space Partitioning Solver



Search Space Partitioning Solver

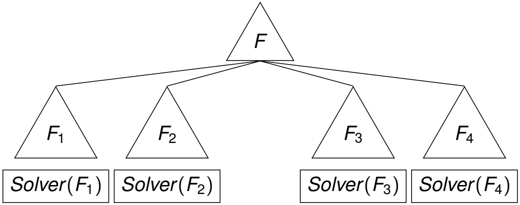


Search Space Partitioning Solver

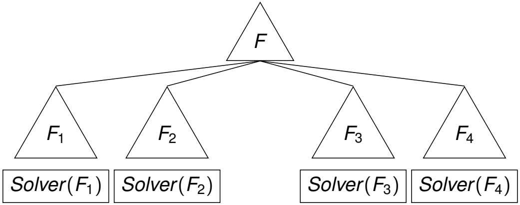


Search Space Partitioning Solver

Non-overlapped solving



Search Space Partitioning Solver

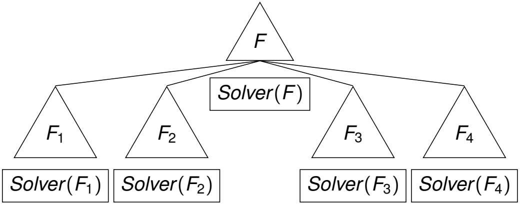


Can not guarantee [Hyvärinen et al., 2009] that:

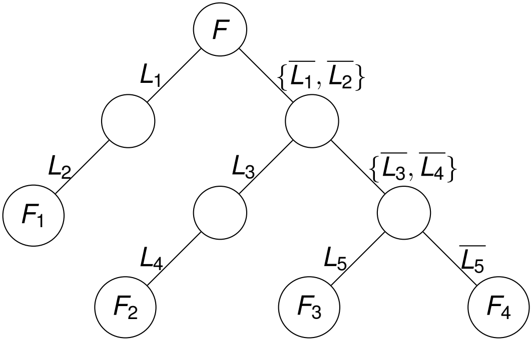
- $max(runtime\ of\ Solver(F_1), Solver(F_2), Solver(F_3), Solver(F_4)) \leq (runtime\ of\ Solver(F))$

Search Space Partitioning Solver

Overlapped solving [Hyvärinen and Manthey, 2012]



Partitioning Function - Tabu Scattering



- $F_1 = \{\{L_1\}, \{L_2\}\} \cup F$
- $F_2 = \{\{\overline{L_1}, \overline{L_2}\}\} \cup \{\{L_3\}, \{L_4\}\} \cup F$
- $F_3 = \{\{\overline{L_1}, \overline{L_2}\}\} \cup \{\{\overline{L_3}, \overline{L_4}\}\} \cup \{\{L_5\}\} \cup F$
- $F_4 = \{\{\overline{L_1}, \overline{L_2}\}\} \cup \{\{\overline{L_3}, \overline{L_4}\}\} \cup \{\{\overline{L_5}\}\} \cup F$

Position-Based Clause Sharing [Lanti and Manthey, 2013]

$$F := \{\{1,2,3\},\{\overline{3},\overline{2}\},\{\overline{2},4,\overline{1}\},\{\overline{2},\overline{4},\overline{1}\},\{4,2,5,\overline{7}\},\{4,2,\overline{5}\}\}$$

$$\{\{7\}\}^{1} \qquad \{\{\overline{7}\}\}^{2}$$

$$\{\ldots,\{\overline{2},\overline{4},\overline{1}\},\{4,2,5\}^{1},\{4,2,\overline{5}\}\}^{1} \qquad F^{2}$$

$$\{\{1\}\}^{11} \qquad \{\{\overline{1}\}\}^{12}$$

$$\{\{\overline{3},\overline{2}\},\{\overline{2},4\}^{11},\{\overline{2},\overline{4}\}^{11},\ldots\}^{11} \qquad \{\{2,3\}^{12},\{\overline{3},\overline{2}\},\ldots\}^{12}$$

$$\{\{3\}\}^{121} \qquad \{\{\overline{3}\}\}^{122}$$

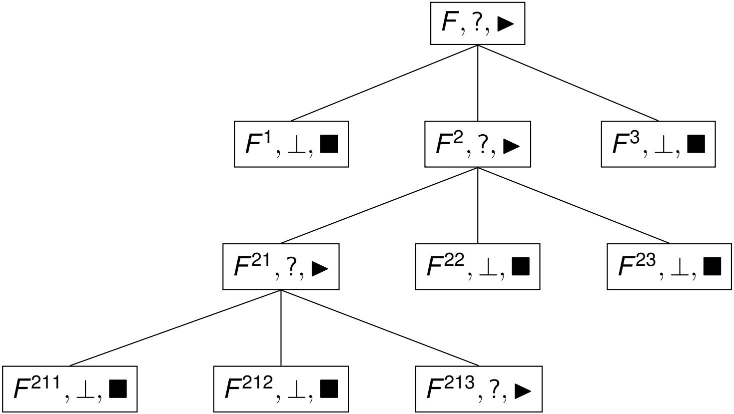
$$F^{121} := \{\{\overline{2}\}^{121},\{4,2,5\}^{1},\{4,2,\overline{5}\}\}^{121} \qquad F^{122}$$

- $\{4,2\} := \{4,2,5\}^{1} \otimes \{4,2,\overline{5}\}^{\epsilon}$
- $\{4,2\}$ is tagged with position 1
- Learnt clause $\{4,2\}^{1}$ is a semantic consequence of nodes of the sub-tree at position 1

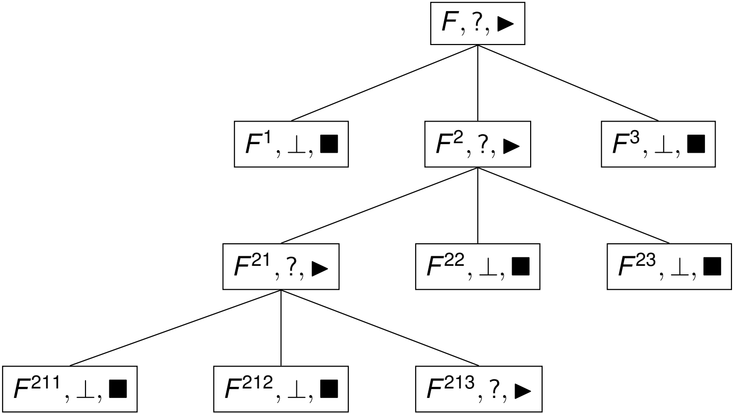
Partition Solving Limit

- Previous works have a restriction on the solving time for each node
- We relax the restriction of the limit on solving time and the intuition behind is: a node is stopped that is close to find the result

Only Child Scenario

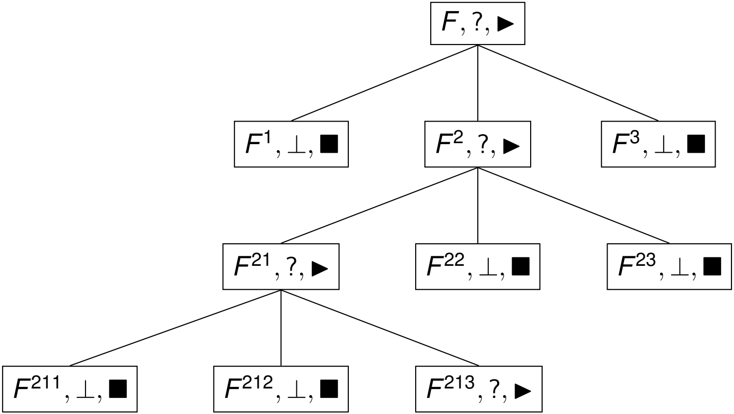


Only Child Scenario



- Parent node is looking in the search space which has been solved by one of its child

Only Child Scenario



- Parent node is looking in the search space which has been solved by one of its child
- Parent node is looking in the search space where its unsolved child is looking

Only Child Scenario

Two ways to look at the only child scenario:

Only Child Scenario

Two ways to look at the only child scenario:

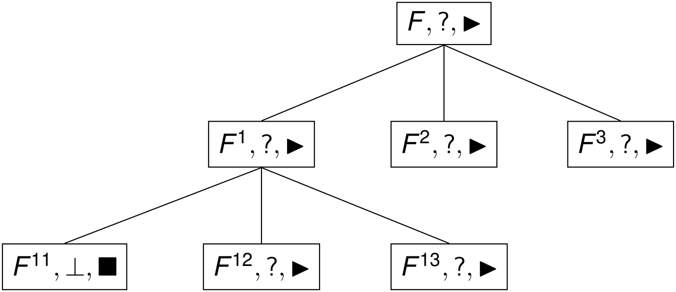
- Avoid
  - Reintroduce solving limit

Only Child Scenario

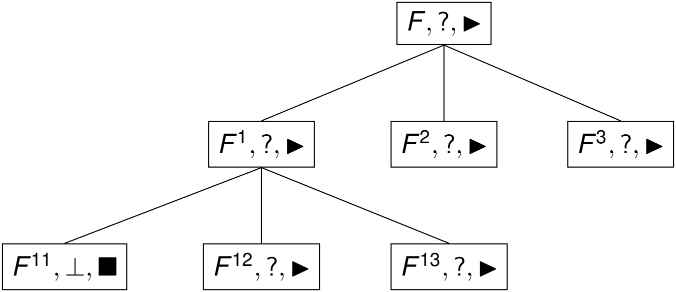
Two ways to look at the only child scenario:

- Avoid
  - Reintroduce solving limit
- Exploit
  - Information sharing

Conflict Driven Node Killing



Conflict Driven Node Killing



- Position of the empty clause obtained for F^{11} is 1
  - Kill the parent node at position 1

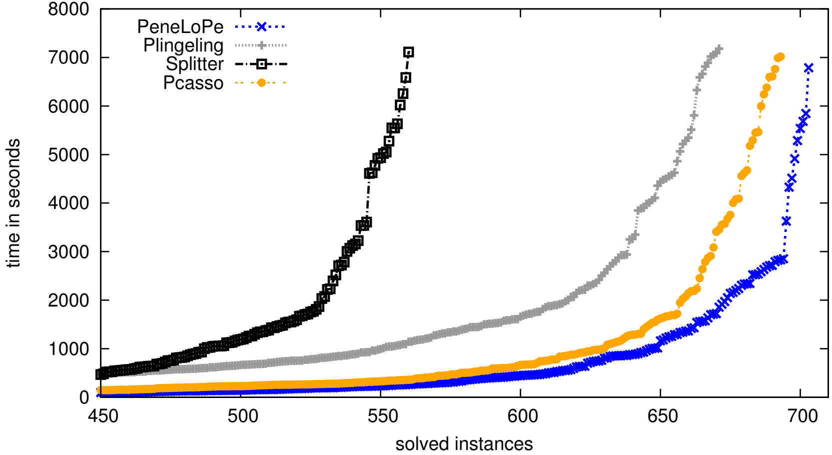
Other Improvements

- Tabu Scattering with Lookahead
- Sorting Partitions per node
- Sharing VSIDS and Phase Saving
- Dynamic Clause Sharing
- Different Restarts
- Different Clause Cleaning

Evaluation Methodology

- Benchmark: 880 instances from SAT Competitions (2009, 2011, 2012)
- Hardware: 16 core AMD Opteron 6274 CPUs with 2.2 GHz
- using 8 cores, 7200 sec walltime and 8 GB memory
- Our Solver PCASSO in comparison with
  - PENELOPE
  - PLINGELING

Evaluation



PeneLoPe
Plingeling
Splitter
Pcasso
time in seconds
solved instances
0
1000
2000
3000
4000
5000
6000
7000
8000
450
500
550
600
650
700

Evaluation

Configuration	Solved	SAT	UNSAT	Median	CPU ratio
Plingeling	672	296	376	442.28	6.38
Pcasso	696	303	393	136.17	6.67
PeneLoPe	704	304	400	89.39	6.90

- Solved: number of solved instances
- SAT: number of satisfiable solved instances
- UNSAT: number of unsatisfiable solved instances
- Median: median solving time and is in seconds
- CPU ratio: ratio of CPU time over real time

Conclusion

- Improved search space partitioning solver
- Comparable in performance with the state-of-the-art portfolio solver
- Best of both worlds

Conclusion

- Improved search space partitioning solver
- Comparable in performance with the state-of-the-art portfolio solver
- Best of both worlds

- Future Work
  - Use of simplification with in search
  - Clause freezing

References I

Hyvärinen, A. E., Junttila, T., and Niemelä, I. (2009).
Partitioning search spaces of a randomized search.
In *Proceedings of the XIth International Conference of the Italian Association for Artificial Intelligence Reggio Emilia on Emergent Perspectives in Artificial Intelligence*, AI*IA '09:, pages 243–252, Berlin, Heidelberg. Springer-Verlag.

Hyvärinen, A. E. J. and Manthey, N. (2012).
Designing scalable parallel SAT solvers.
In Cimatti, A. and Sebastiani, R., editors, *SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 214–227. Springer.

Lanti, D. and Manthey, N. (2013).
Sharing information in parallel search with search space partitioning.
Technical report, Dresen University of Technology.