Optimal Parallel Task Scheduling via SAT

Matthew Akram
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
Karlsruhe, Germany
mazfh85246@gmail.com

Dominik Schreiber
Institute of Theoretical Informatics
Karlsruhe Institute of Technology
Karlsruhe, Germany
dominik.schreiber@kit.edu

***Abstract*—We introduce SAT benchmark instances based on the optimal parallel task scheduling problem $P||C_{\max}$.**
***Index Terms*—SAT solving, scheduling algorithms**

I. Introduction

The strongly NP-hard [9] scheduling problem $P||C_{\max}$ is concerned with mapping a set of n tasks with known execution times $W = \{w_1, \ldots, w_n\} \in \mathbb{N}_+^n$ to a set of m identical machines such that the overall completion time $C_{\max}$ is minimized [10]. State-of-the-art optimal approaches are branch-and-bound schemes [3, 7] and ILP translations [16].

While reductions to SAT have previously shown promise for numerous problems from automated planning [12, 18, 19] and scheduling [5, 6, 14, 23], these works include rule sets that impose significant logical constraints on feasible solutions (e.g., causal dependencies or pre– and postconditions w.r.t. a world state). By contrast, we try a similar approach for the unconstrained scheduling problem $P||C_{\max}$. In this report, we outline our encoding on an abstract level—the exact specification can be found in the 1st author's master thesis [2].

II. Encoding

We sort our jobs $j_1, \ldots, j_n$ by duration in decreasing order, i.e., $w_1 \geq \ldots \geq w_n$. We denote the assignment of job j_i to processor p_x as $A_i = x$. We refer to the *load* of a processor p_x as $C_x = \sum_{i:A_i=x} w_i$. The value to minimize is the completion time or *makespan* $C_{\max} = \max_x\{C_x\}$.

In order to represent the assignment of job j_i to processor p_x, we introduce a Boolean variable $a_{i,x}$. The ordering of jobs on a certain processor is irrelevant and does not need to be represented. We ensure that each job j_i is assigned to exactly one processor with a clause $(\bigvee_x a_{i,x})$ and $\mathcal{O}(m^2)$ clauses of the shape $(\neg a_{i,x} \vee \neg a_{i,x'})$.[1] This results in a total of $O(n \cdot m)$ variables and $O(n \cdot m^2)$ clauses to represent the assignments of jobs to processors. While asymptotically more efficient encodings exist [22], we found that the simple encoding suffices for most feasible instances. Finally, we ensure that the sum of the weights assigned to each processor does not exceed the considered upper bound U, which is equivalent to the *Pseudo-Boolean Constraint* (PBC)

$$a_{1,x} \cdot w_1 + a_{2,x} \cdot w_2 + \ldots + a_{n,x} \cdot w_n \leq U$$

[1]The latter constraints are not actually required and omitting them results in a satisfiability-equivalent problem. However, we found these constraints to be useful since they drastically reduce search space.

for each processor p_x. Among multiple established methods to encode PBCs into SAT [8, 15, 17, 22], we found encoding the PBC into a *Reduced Ordered Binary Decision Diagram* (BDD) and then encoding the BDD into SAT [1] to be the most promising approach. Simply put, our BDD is a directed acyclic graph with a single source node and two outgoing edges per node, each representing a yes-or-no decision, where all paths eventually end in `true` or `false`. The diagram represents which combinations of jobs can be assigned to a processor p_x such that $C_x \leq U$. Moreover, BDDs can be reduced by merging isomorphic subgraphs and by eliminating nodes whose decisions are inconsequential. In our case, these reductions provide useful insights since they back-propagate the hard constraint $C_x \leq U$ through the sequences of decisions and detect infeasible and/or superfluous job combinations. In addition, we apply some pruning rules to the problem in the shape of further constraints: We force the i-th unassigned job onto one of the first i processors of identical load, and we disallow assigning j_{i+1} to the first processor which can fit j_i but not j_i and j_{i+1}. We also force j_i onto p_x whenever p_x has w_i space remaining and j_i is unassigned. Note that these rules require to reference the reachable values of C_x after the first i jobs are already assigned, which our BDD-based encoding is designed to allow. For details on our encoding and the arguments of soundness for the underlying pruning rules, we refer to the 1st author's master's thesis [2].

III. Benchmarks

We considered instances by Lawrinenko [13] (inspired by Haouari and Jemmali [11]; summarized and extended by Mrad and Souayah [16]), which are difficult due to combinatorial properties. These instances feature $n \in [20, 220]$ and $n/m \in [2, 3]$ with a number of different distributions of job sizes. We first ran our SAT-based $P||C_{\max}$ scheduling approach on all instances with a timeout of 500 s (using KissAT [4] as a SAT solver) and only consider formulas at which the SAT solver timed out. We then attempted to solve a random selection of the remaining instances using 1216 cores (16 nodes) of the HoreKa HPC cluster with MallobSat [20] with ImpCheck [21], i.e., with on-the-fly LRAT checking enabled. We thus consider the results to be reliable. We selected all 11 instances found to be unsatisfiable and added nine modestly difficult satisfiable instances (running times ranging between 13 s and 54 s).

Result	Name	m
SAT	pcmax-scheduling-m12-8049-55035-SAT.cnf	12
	pcmax-scheduling-m15-2352-13561-SAT.cnf	15
	pcmax-scheduling-m23-6057-43824-SAT.cnf	23
	pcmax-scheduling-m24-17855-226744-SAT.cnf	24
	pcmax-scheduling-m24-24102-255206-SAT.cnf	24
	pcmax-scheduling-m35-32274-371389-SAT.cnf	35
	pcmax-scheduling-m37-28831-324346-SAT.cnf	37
	pcmax-scheduling-m40-26287-324155-SAT.cnf	40
	pcmax-scheduling-m43-38782-385402-SAT.cnf	43
UNSAT	pcmax-scheduling-m11-1517-6802-UNSAT.cnf	11
	pcmax-scheduling-m13-1655-9604-UNSAT.cnf	13
	pcmax-scheduling-m13-2011-12813-UNSAT.cnf	13
	pcmax-scheduling-m13-2826-11437-UNSAT.cnf	13
	pcmax-scheduling-m14-3640-14524-UNSAT.cnf	14
	pcmax-scheduling-m19-2974-16501-UNSAT.cnf	19
	pcmax-scheduling-m19-10199-62102-UNSAT.cnf	19
	pcmax-scheduling-m20-3726-23254-UNSAT.cnf	20
	pcmax-scheduling-m20-4135-27005-UNSAT.cnf	20
	pcmax-scheduling-m26-6398-62377-UNSAT.cnf	26
	pcmax-scheduling-m30-14113-167638-UNSAT.cnf	30

TABLE I
SELECTION OF SUBMITTED BENCHMARKS WITH THE RESULT CLAIMED BY MALLOBSAT WITH IMPCHECK.

Table I lists the selection of submitted benchmark instances for future reference. The value of m, the number of variables and clauses, and the result are part of the file name.

ACKNOWLEDGMENT

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 882500).

This work was performed on the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research.

REFERENCES

[1] I. Abío et al. "BDDs for Pseudo-Boolean Constraints – Revisited". In: *Theory and Applications of Satisfiability Testing - SAT 2011*. Ed. by Karem A. Sakallah and Laurent Simon. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 61–75. ISBN: 978-3-642-21581-0.

[2] Matthew Akram. *SAT Techniques for Multiprocessor Scheduling on Uniform Machines*. Karlsruhe Institute of Technology. 2024.

[3] Matthew Akram et al. "Engineering Optimal Parallel Task Scheduling". To be published. 2024.

[4] Armin Biere, Mathias Fleury, and Florian Pollitt. "CaDiCaL_vivinst, IsaSAT, Gimsatul, Kissat, and TabularaSAT Entering the SAT Competition 2023". In: *SAT Competition*. 2023, p. 14.

[5] James M Crawford and Andrew B Baker. "Experimental results on the application of satisfiability algorithms to scheduling problems". In: *AAAI*. Vol. 2. 1994, pp. 1092–1097.

[6] Steve Dai, Gai Liu, and Zhiru Zhang. "A scalable approach to exact resource-constrained scheduling based on a joint SDC and SAT formulation". In: *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2018, pp. 137–146.

[7] M. Dell'Amico et al. "Heuristic and Exact Algorithms for the Identical Parallel Machine Scheduling Problem". In: *INFORMS Journal on Computing* 20 (Aug. 2008), pp. 333–344. DOI: 10.1287/ijoc.1070.0246.

[8] N. Eén and N. Sörensson. "Translating Pseudo-Boolean Constraints into SAT." In: *J. Satisf. Boolean Model. Comput.* 2.1-4 (2006), pp. 1–26. URL: http://dblp.uni-trier.de/db/journals/jsat/jsat2.html#EenS06.

[9] M.R. Garey and D.S. Johnson. "Strong NP-completeness results". In: *J. ACM* 25.3 (1978), pp. 499–508. DOI: 10.1145/322077.322090.

[10] Ronald Lewis Graham et al. "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Annals of discrete mathematics*. Vol. 5. Elsevier, 1979, pp. 287–326. DOI: 10.1016/S0167-5060(08)70356-X.

[11] Mohamed Haouari and Mahdi Jemmali. "Tight bounds for the identical parallel machine-scheduling problem: Part II". In: *Int. Trans. Operational Research* 15.1 (2008), pp. 19–34. DOI: https://doi.org/10.1111/j.1475-3995.2007.00605.x.

[12] Henry Kautz, David McAllester, Bart Selman, et al. "Encoding plans in propositional logic". In: *KR* 96 (1996), pp. 374–384.

[13] Alexander Lawrinenko. "Identical parallel machine scheduling problems: structural patterns, bounding techniques and solution procedures". en. PhD thesis. Jena: Friedrich-Schiller-Universität, 2017. URL: https://www.db-thueringen.de/receive/dbt_mods_00032188.

[14] Alexandre Lemos et al. "Iterative Train Scheduling under Disruption with Maximum Satisfiability". In: *Journal of Artificial Intelligence Research* 79 (2024), pp. 1047–1090.

[15] N. Manthey, T. Philipp, and P. Steinke. "A More Compact Translation of Pseudo-Boolean Constraints into CNF Such That Generalized Arc Consistency Is Maintained". In: *KI 2014: Advances in Artificial Intelligence*. Ed. by Carsten Lutz and Michael Thielscher. Cham: Springer International Publishing, 2014, pp. 123–134. ISBN: 978-3-319-11206-0.

[16] Mehdi Mrad and Nizar Souayah. "An Arc-Flow Model for the Makespan Minimization Problem on Identical Parallel Machines". In: *IEEE Access* 6 (2018), pp. 5300–5307. DOI: 10.1109/ACCESS.2018.2789678.

[17] T. Philipp and P. Steinke. "PBLib – A Library for Encoding Pseudo-Boolean Constraints into CNF". In: *Theory and Applications of Satisfiability Testing – SAT 2015*. Ed. by Marijn Heule and Sean Weaver. Cham: Springer International Publishing, 2015, pp. 9–16. ISBN: 978-3-319-24318-4.

[18] Jussi Rintanen. "Madagascar: Scalable planning with SAT". In: *Proc. International Planning Competition*. 2014.

[19] Dominik Schreiber. "Lilotane: A Lifted SAT-Based Approach to Hierarchical Planning". In: *Journal of Artificial Intelligence Research (JAIR)* 70 (2021), pp. 1117–1181. DOI: 10.1613/jair.1.12520.

[20] Dominik Schreiber. "Scalable SAT Solving and its Application". PhD thesis. Karlsruhe Institute of Technology, 2023. DOI: 10.5445/IR/1000165224.

[21] Dominik Schreiber. "Scalable Trusted SAT Solving with on-the-fly LRAT Checking". Submitted to *Int. Conf. on Theory and Applications of Satisfiability Testing (SAT)*. 2024.

[22] Carsten Sinz. "Towards an optimal CNF encoding of boolean cardinality constraints". In: *Proc. CP*. Springer. 2005, pp. 827–831. DOI: 10.1007/11564751_73.

[23] Hantao Zhang, Dapeng Li, and Haiou Shen. "A SAT Based Scheduler for Tournament Schedules." In: *SAT*. 2004.