Resource-constraint project scheduling with task preemption and setup times by heuristically augmented SAT solver

Jasper Vermeulen[1]
Supervisor: Emir Demirović[1]
[1]EEMCS, Delft University of Technology, The Netherlands
J.Vermeulen-1@student.tudelft.nl

Abstract

TBD in final paper

1 Introduction

The problem of scheduling tasks arises in industries all the time. It is not hard to imagine that generating an optimised schedule can be of great profit for production or logistic operations. For example, optimisation can minimise the overal required time or minimise the delay before starting a task. Because this type of problem is so prevalent it has already been subject to much research.

Formally this specific type of problem is known as the resource-constrained project scheduling problem (RCPSP). The standard RCPSP has a set of tasks that each require a specified resource for a given amount of time and precedence restrictions are given for the tasks. On its own, this problem definition for standard RCPSP is limited and of little use to realistic applications where additional constraints must be followed or more options are available. To make sure the researched algorithms solving the scheduling problem would have a wider use case, many variations and extensions to the problem definition have been classified over time [1], [2]. More recently, the variations and extensions have also been surveyed and put into a structured overview [3].

For this research, the preemptive resource-constrained project scheduling problem with setup times (PRCPSP-ST) variant is under study. Preemption allows an activity to be interrupted during its scheduled time by another activity. Each interruption can be seen as splitting the activity into multiple smaller activities. The setup times are introduced for each interruption in an activity to discourage endless splits resulting in a chaotic schedule. Both a model for allowing preemption [4] and including setup times [5] have already been established. Both models have been combined and a proposed algorithm for it was found to result in a reduction of the makespan compared to the optimal schedule without activity preemption [6]. Within this algorithm, the activities are split into all possible integer time segments and a SAT solver makes a selection from these segments [7]. The resulting list was used to construct a schedule with a genetic algorithm established in earlier research [8].

In the research done on solving RCPSP variants, the focus has been on heuristic and meta-heuristic algorithms. These algorithms are usually variants of branch-and-bound algorithm [4] or a form of genetic algorithms [9] that were established for the standard RCPSP.

The use of SAT solvers as a part of the algorithm was mentioned earlier but using a SAT solver as a complete solution to try and solve PRCPSP-ST instances has not been researched before. A SAT solver is a very general tool that can be used on any algorithmic problem as long as it is encoded as the required input for the solver. Heuristic algorithms are specialised by knowledge or an insight into the problem variation translated into a heuristic rule for the algorithm to use. Because the RCPSP is known to be strongly NP-hard [10], the SAT solver might not be more efficient than the heuristic and meta-heuristic at first. But, a SAT solver can also be modified to include a heuristic for the PRCPSP-ST variant. In a way, the addition of a heuristic can be seen as an augmentation of the SAT solver. The augmentation is making the solver more specialised for the PRCPSP-ST variant and could result in a better performance than other algorithms.

Because there is room to try and find out if a SAT solver can outperform the heuristic and meta-heuristic algorithms the main research question is: *Can the addition of a simple heuristic to a SAT solver algorithm used to solve to PRCPSPST models reduce the average makespan of the resulting schedule when run for an equal amount time?*. The expectation for this research is to first show that a SAT solver can be used to solve PRCPSP-ST instances. And next, show that the heuristically augmented version of the SAT solver algorithm will result in a lower makespan than a heuristic algorithm when running an equal amount of time.

Report section structure goes here

2 Problem Description

2.1 Formal Problem Definition

3 Heuristics and augmentation of SAT solvers

To evaluate the performance of a heuristically augmented SAT solver on PRCPSP-ST instances comparative data needs to be generated. To make sure the variables between algorithms are limited they were all designed, implemented and run on the same hardware for this research. This way the at least coding skills and hardware improvements over time are not introduced in the data. Improved performance of new algorithms only as a result of the use of more recent hardware has shown up in the past as described in section 5. Three algorithms are used to solve the same problem instances for an equal amount of time: a heuristic algorithm, SAT solver, heuristically augmented SAT solver.

The heuristic algorithm is an adapted version of the iterated greedy algorithm [11]. It was designed for flow-shop scheduling but with a few tweaks it can also be applied to RCPSP.

For a SAT solver to be used the PRCPSP-ST has to be encoded into conjunctive normal form boolean logic. When the encoding is made any SAT solver is able to provide feasible schedules. Because there is a clear objective to reduce the makespan a more advanced MAX-SAT solver is used. This will create feasible schedules while also trying to optimize to an objective function.

To augment a SAT solver for a specific problem the code of the solver has to be changed. In the case of this research a heuristic function will be added that improves the selection of a possible variable. The heuristic function will use knowledge about PRCPSP-ST to try and select a variable that will reduce the resulting schedule makespan.

3.1 Heuristic

3.2 CNF Encoding

Variable $w_{i,x,y,t}$ is equal to 1 if activity segment $S^i_x D_y$ starts at time t, and 0 otherwise. Variable $c_{i,x,y,s}$ is 1 if segment $S^i_x D_y$ contains $S^i_s D_1$ as a segment, and 0 otherwise.

$$\neg c_{i,x,y,s} \bigvee_{t=0,...,T-y} w_{i,x,y,t} \tag{1}$$

$$\forall i \in N; x \in 1,...,d_i; y \in 1,...,d_i - x + 1; s \in 1,...,d_i$$

Variable $u_{i,x,y,t}$ is equal to 1 if activity segment $S^i_x D_y$ is in process at time t, and 0 otherwise.

$$\neg w_{i,x,y,t} \vee u_{i,x,y,l} \tag{2}$$

$$\forall i \in N; x \in 1,...,d_i; y \in 1,...,d_i - x + 1;$$
$$t \in 0,...,T-y; l \in t,...,t+d_i-1$$

$$\neg w_{i,x,y,t} \forall (i,x,d),j,k,l)) \in E; t \in 0,...,T-d_j (3)$$

3.3 SAT solver

4 Experimental Setup and Results

5 Responsible Research

6 Discussion

7 Conclusions and Future Work

References

[1] W. Herroelen, E. Demeulemeester, and B. De Reyck, *A Classification Scheme for Project Scheduling*. Boston, MA: Springer US, 1999, pp. 1–26. [Online]. Available: https://doi.org/10.1007/978-1-4615-5533-9_1

[2] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221798002045

[3] S. Hartmann and D. Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 207, no. 1, pp. 1–14, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221709008558

[4] E. Demeulemeester and W. Herroelen, "An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 90, no. 2, pp. 334–348, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377221795003584

[5] L. Kaplan, "Resource-constrained project scheduling with setup times," 1991.

[6] M. Vanhoucke and J. Coelho, "Resource-constrained project scheduling with activity splitting and setup times," *Computers Operations Research*, vol. 109, pp. 230–249, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054819301170

[7] J. Coelho and M. Vanhoucke, "Multi-mode resource-constrained project scheduling using rcpsp and sat solvers," *European Journal of Operational Research*, vol. 213, no. 1, pp. 73–82, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037722171100230X

[8] D. Debels and M. Vanhoucke, "A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem," *Operations Research*, vol. 55, no. 3, pp. 457–469, 2007. [Online]. Available: http://www.jstor.org.tudelft.idm.oclc.org/stable/25147093

[9] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics (NRL)*, vol. 45, no. 7, pp. 733–750, 1998. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1520-6750(199810)45:7%3C733::AID-NAV5%3E3.0.CO;2-C

[10] J. Blazewicz, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Scheduling subject to resource constraints: classification and complexity," *Discret. Appl. Math.*, vol. 5, pp. 11–24, 1983.

[11] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221705008507