Solving a Multi-resource Partial-ordering Flexible Variant of the Job-shop Scheduling Problem with Hybrid ASP (Extended Abstract) [*]

Giulia Francescutto
Siemens AG Österreich, Vienna, Austria
giulia.francescutto@siemens.com

Konstantin Schekotihin
University of Klagenfurt, Klagenfurt, Austria
konstantin.schekotihin@aau.at

Mohammed M. S. El-Kholany
University of Klagenfurt, Klagenfurt, Austria
Cairo University, Cairo, Egypt
mohammed.el-kholany@aau.at

In many industries, scheduling is a key component to efficient management of resources and, thus, ensuring competitiveness and success of companies by reducing the consumption of time and money. In this work, we propose a *Multi-resource Partial-ordering Flexible Job-shop Scheduling* formulation to capture scheduling scenarios where partially-ordered sequences of operations must be scheduled on multiple required resources, such as tools and specialists. The resources are flexible and can process one or more kind of operations based on their characteristics. We have built a model using Answer Set Programming in which the execution time of operations is determined using Difference Logic. Furthermore, we introduced two multi-shot strategies that identify the time bounds and find a schedule while minimizing the total tardiness. Experiments on real-world instances from an Infineon Fault Analysis lab show that the devised model yields optimized schedules for 87 out of 91 instances.

1 Introduction

Job-shop Scheduling (JSS) [7] is one of the most well-known scheduling problems in which a set of machines are used to execute given jobs, represented as sequences of operations, and the goal is to complete the jobs as soon as possible. Extensions like flexible [1] and multi-resource [2] JSS generalize the allocation of a machine and additional resources for an operation to suit practical scheduling applications.

In this work, we consider a *Multi-resource Partial-ordering Flexible JSS* (MPF-JSS) problem, where multiple resources are needed for executing operations, and jobs consist of partially-ordered sequences of operations to be completed. More specifically, we distinguish machines and engineers as two separate sets of resources required to perform an operation. Flexibility lies in the choice among several resources with the required skills in each category. The goal is to determine the best possible execution order and allocation of operations to resources for minimizing the total tardiness of jobs wrt. their deadlines.

We model the MPF-JSS problem by an encoding in Answer Set Programming (ASP) with Difference Logic (DL) [4] and take advantage of multi-shot solving [5]. DL constraints allow for compactly expressing timing requirements, avoiding grounding issues that would otherwise be encountered when

[*]This work was done during an internship of Giulia Francescutto at Infineon Technologies Austria AG and has also been presented at the 17th European Conference on Logics in Artificial Intelligence (JELIA 2021) [3]. Mohammed M. S. El-Kholany was supported by KWF project 28472, cms electronics GmbH, FunderMax GmbH, Hirsch Armbänder GmbH, incubed IT GmbH, Infineon Technologies Austria AG, Isovolta AG, Kostwein Holding GmbH, and Privatstiftung Kärntner Sparkasse.

Submitted to:
ICLP 2022

a feasible schedule necessitates a large number of time points. By means of multi-shot solving, we implement two strategies to identify upper time bounds on feasible schedules whose tardiness is further subject to minimization. We tested our proposed model on a dataset representing ten operational days of an Infineon Fault Analysis lab. Our experiments yield success to optimize schedules for 87 out of 91 real-world instances, while only a very small number of jobs can be handled without multi-shot solving.

2 Modeling MPF-JSS with Hybrid ASP

We consider a scheduling problem in which different resources are interrelated to process the upcoming operations. In particular, MPF-JSS is an extension of the classical JSS problem where three additional aspects are considered: *(a) Multi-resources* – several categories of resources can be required to execute an operation; *(b) Partially-ordered* – some operations cannot be executed before completing their predecessors and others are unaffected by such constraints; and *(c) Flexible* – an operation can be performed by a variety of available resources. More specifically, an MPF-JSS instance consists of a set of jobs, characterized by a partially ordered set of operations to be performed, where each operation needs to be allocated to some machine and (possibly) an engineer with the required skills. The following constraints must be respected: *(i)* once an operation starts, it cannot be interrupted; *(ii)* each resource can perform only one operation at a time; *(iii)* the execution times of operations of the same job cannot overlap; and *(iv)* operations must be scheduled according to the partial order given by jobs.

We propose two multi-shot ASP modulo DL solving strategies. The first approach incrementally increases the upper bound on the tardiness of each job in order to identify a limit for which schedules are feasible. That is, search starts with the tardiness bound 0, and if this yields unsatisfiability (UNSAT), the bound is incremented by a constant, set by a parameter of our Python control script on top of *clingo*[DL] [6], and this process proceeds until some schedule is found. Figure 1 illustrates an execution of the incremental search strategy, where the bound on jobs' tardiness is increased in steps of size 2 until satisfiability (SAT) is obtained with the time bound 8. ASP optimization methods are then used to find a schedule minimizing the total tardiness, i.e., the sum of delays over jobs completed after their deadlines.

The second approach performs an exponential binary search for the exact tardiness bound at which schedules get feasible. As displayed in Figure 2, the increment on the bound is gradually increased up to step size 4 for finding the first schedule with the time bound 8. A binary search scheme then successively halves the step size for converging to the exact bound, 7 in this case, for which schedules are feasible, whose total tardiness is subject to ASP optimization in the last phase of the multi-shot solving process.

3 Results

We conducted experiments on a set of real-world instances of the MPF-JSS problem representing the operations at ten days randomly selected from the work records of an Infineon Fault Analysis lab. The instances yield the following approximate number of components based on the dimensions of the studied lab: *(i)* 50 operation kinds; *(ii)* 75 machines; and *(iii)* 45 engineers. For each of the selected days, the number of jobs ranges between 30 and 50. We further split each instance into up to ten sub-instances by picking 5, 10, 15, 20, etc. of the respective jobs, which resulted in a total of 91 instances of varying size.

For each obtained instance, we ran three solving strategies with *clingo*[DL] (version 1.1.0): *(1) single-shot* – standard ASP modulo DL program with a tardiness bound precomputed using a heuristic; *(2) inc* – the incremental multi-shot variant with a constant step size of 20 time points; and *(3) exp* – the

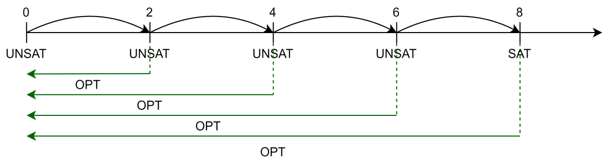

Figure 1: Incremental approach

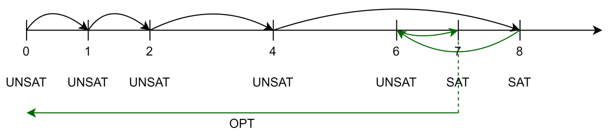

Figure 2: Exponential approach

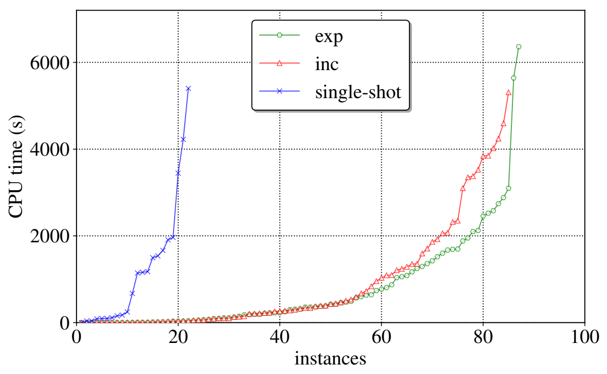

Figure 3: Cactus plot of solving times

exponential multi-shot solving approach. We restricted each solver run to 2 hours wall clock time on an Intel 3930K workstation with 64GB RAM under Ubuntu 18.05.

Figure 3 shows a cactus plot comparing the solving performance of the two multi-shot approaches and the single-shot version as baseline. The latter times out on all instances with more than 15 jobs and merely succeeds to optimize schedules on instances with 15 jobs for two of the ten days. The two multi-shot approaches significantly outperform the single-shot version. While the incremental strategy yields optimized schedules for 85 out of 91 instances, the exponential multi-shot variant scales up to 87 instances. This advantage is due to tighter tardiness bounds identified by the exponential strategy, thus reducing the efforts for ASP optimization to minimize the total tardiness. Given that a lower tardiness bound is more restrictive, the incremental variant manages to find better schedules than the exponential strategy for three instances, where the total tardiness improvement amounts to 80 time points on average.

References

[1] Peter Brucker & Rainer Schlie (1990): *Job-shop scheduling with multi-purpose machines*. *Computing* 45(4), pp. 369–375.

[2] Stéphane Dauzère-Pérès, W. Roux & Jean Lasserre (1998): *Multi-resource shop scheduling with resource flexibility*. *EJOR* 107(2), pp. 289–305.

[3] Giulia Francescutto, Konstantin Schekotihin & Mohammed El-Kholany (2021): *Solving a multi-resource partial-ordering flexible variant of the job-shop scheduling problem with hybrid ASP*. In: *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA 2021)*, Springer, pp. 313–328.

[4] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub & Philipp Wanko (2016): *Theory solving made easy with clingo 5*. In: *Technical Communications of the International Conference on Logic Programming (ICLP 2016)*, Leibniz International Proceedings in Informatics, pp. 2:1–2:15.

[5] Martin Gebser, Roland Kaminski, Benjamin Kaufmann & Torsten Schaub (2019): *Multi-shot ASP solving with clingo*. *TPLP* 19(1), pp. 27–82.

[6] Tomi Janhunen, Roland Kaminski, Max Ostrowski, Sebastian Schellhorn, Philipp Wanko & Torsten Schaub (2017): *Clingo goes linear constraints over reals and integers*. *TPLP* 17(5-6), pp. 872–888.

[7] Selmer Martin Johnson (1954): *Optimal two-and three-stage production schedules with setup times included*. *Naval Research Logistics Quarterly* 1(1), pp. 61–68.