Energy-Aware Double-Flexible Job Shop Scheduling with CP and ASP modulo CSP

Francesco Zuccato[1], Patrick Rodler[1], Gerhard Friedrich[1], Konstantin Schekotihin[1], Giray Havur[2], and Richard Comploi-Taupe[2]

[1] University of Klagenfurt, Klagenfurt, Austria
[2] Siemens AG Österreich, Vienna, Austria

Abstract. We present a comparison of two declarative approaches, Answer Set Programming (ASP) and Constraint Programming (CP), for encoding and solving the Energy-Aware Double-Flexible Job Shop Scheduling Problem (E-DFJSP) with alternative machine modes as well as setup, and transport operations. Our evaluation compares IBM CP Optimizer, a state-of-the-art solver optimized for scheduling, with clingcon, a general-purpose hybrid ASP solver enriched with CSP constraints.

Keywords: Flexible Job Shop Scheduling · Energy-Aware Scheduling · Answer Set Programming · Constraint Programming

1 Introduction

Energy-efficient scheduling is crucial for sustainable manufacturing, but real-world implementations remain challenging due to the NP-hard nature of the Job Shop Scheduling Problem (JSP) [7,4]. Thanks to their declarative approach, Answer Set Programming (ASP) and Constraint Programming (CP) are particularly suitable for solving the JSP, and [3,2] proved their effectiveness even for large-scale instances. We compare these paradigms for the *Energy-Aware Double-Flexible JSP (E-DFJSP)* [8], which extends the Flexible JSP [1] with *double flexibility* (two types of flexible resources, i.e., machine and worker), *machine modes* (time/energy trade-offs), and *setup* and *transport operations*. The objective is to minimize *(i)* tardiness T, *(ii)* energy consumption E, and *(iii)* makespan C_{max}. The study is motivated by a steel-cutting use case, where each job involves cut operations that require assigning a machine, a worker, and a machine mode $\langle V_f, V_c \rangle$, where V_f is the feed rate and V_c the blade speed.

2 Approach

ASP Model. We use `clingcon` [6] to handle large-domain temporal variables while avoiding the grounding bottleneck. In particular, *(i)* the machines' and workers' assignments, and *(ii)* operations' order are encoded with plain ASP choice rules, while the precedence constraints use linear inequalities.

CP Model. We employ IBM CP Optimizer (`CPO`) [5], which is a state-of-the-art solver for scheduling problems [2]. Operations can be represented with *optional interval variables* (to encode machine and machine modes flexibility), while the resource assignments are modeled by means of `alternative` constraints. Constraint `noOverlap`(r) ensures no overlap among the operations on resource r. Precedence constraints on operations are defined using `endBeforeStart`(o_1, o_2).

3 Evaluation

We compare the performance of `clingcon` and `CPO` on E-DFJSP instances, in terms of time, memory, and solution quality. To assess the latter, we present a greedy algorithm that serves as a baseline.[3]

Dataset Generation. The dataset is generated according to the following parameters: jobs $|J| \in \{50, 100, 200\}$; five machines-workers combinations $\langle |M|, |W| \rangle$[4], assigned workers per machine (2, fixed), machine modes $|V_f| \times |V_c| \in \{2\times2, 5\times5, 10\times10\}$. Jobs have up to 10 operations, and uniformly distributed deadlines. Energy consumption is machine mode specific. For details, see [8].

Experimental Design. We run `clingcon` v5.2.1 and `CPO` v22.1.2 on Ubuntu 22.04 (x86_64), Intel Xeon Gold 6230 (20 cores, 2.10 GHz), 128 GB RAM, with 4 parallel cpus and timeouts $t_{\text{out}} = 600$s. For both solvers, we optimize one goal at a time instead of specifying an explicit lexicographic order (as it is not supported in `clingcon` 5) or a weighted sum (as it is not equivalent to the latter). In order to speed up the default linear convergence of `clingcon`, we implement a binary search to tighten the upper bounds of the objectives. For `CPO`, it is not needed, as it can estimate the bounds internally [5]. We collect: resources' usage (memory $[MB]$, time $[s]$), objective values (T, E, C_{max}), flags (Solved, Optimal), and the number of optimized goals $(\#OG)$.

Preliminary Results. In the initial experiments, similarly to the results in [8], `CPO` can find a solution—of good quality—even for large instances, thanks to specialized scheduling constructs and propagation. `clingcon` can also find an initial solution in many cases, but the solution quality is comparatively poor, as it tends to assign operations over the entire horizon, and then slowly (linearly) improves the solution quality. For this reason, we implemented a binary search on the upper bounds to speed up convergence towards solutions of higher quality. The greedy baseline provides feasible solutions within seconds, confirming its utility as a warm-start generator.

Future Work. We plan to explore portfolio-based methods leveraging both solvers' complementary strengths, investigate warm-starting strategies using greedy solutions, evaluate `clingo-dl` with custom binary search (as it lacks native optimization), and extend the evaluation to larger-scale instances with additional flexibility dimensions and more complex precedence constraints.

[3] The greedy approach uses the Earliest-Deadline-First (EDF) rule to generate feasible schedules by dispatching operations as resources become available.

[4] In order to ensure constant *jobs-per-machine* and *jobs-per-worker* ratios across the dataset, the values of $\langle |M|, |W| \rangle$ depend on the job size $|J|$.

Acknowledgements This work was supported by FFG, contract FO999910235. This research was funded in part by the Austrian Science Fund (FWF) 10.55776/COE12.

References

1. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990)
2. Da Col, G., Teppan, E.C.: Industrial-size job shop scheduling with constraint programming. Operations Research Perspectives **9**, 100249 (2022)
3. El-Kholany, M.M., Gebser, M., Schekotihin, K.: Problem decomposition and multi-shot ASP solving for job-shop scheduling. Theory and Practice of Logic Programming **22**(4), 623–639 (2022)
4. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 29. wh freeman New York (2002)
5. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: Ibm ilog cp optimizer for scheduling: 20+ years of scheduling with constraints at ibm/ilog. Constraints **23**(2), 210–250 (2018)
6. Ostrowski, M., Schaub, T.: Asp modulo csp: The clingcon system. Theory and Practice of Logic Programming **12**(4-5), 485–503 (2012)
7. Pervaiz, S., Kannan, S., Deiab, I., Kishawy, H.: Role of energy consumption, cutting tool and workpiece materials towards environmentally conscious machining: a comprehensive review. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture **234**(3), 335–354 (2020)
8. Zuccato, F., Rodler, P., Friedrich, G., Schekotihin, K., Comploi-Taupe, R.: An energy-aware double-flexible job shop scheduling for an industrial real-world application with constraint programming. In: ECAI Workshop on AI-based Planning for Complex Real-World Applications (CAIPI'25) (2025), (to appear)