ISBN: 978-628-96613-3-0. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2026.1.1.1134

Benchmarking Constraint Programming software: Insights from the Job Shop Scheduling Problem

Francisco Yuraszeck[1]; Daniel Rossit[2,3]; Milenko Milović[4]; Alejandro Córdova[4]; Gabriel Olivares[4]
[1]Department of Engineering Sciences, Universidad Andres Bello, Chile, *francisco.yuraszeck@unab.cl*
[2]Engineering Department, Universidad Nacional del Sur, Argentina
[3]INMABB, CONICET, Argentina
[4]Faculty of Engineering, Universidad Andres Bello, Chile

Abstract: ***In this article, we benchmark three popular constraint programming (CP) solvers for the job shop scheduling problem (JSSP) under identical hardware and computation-time conditions, with the objective of minimizing the makespan. For evaluation purposes, we use 80 classic Taillard instances. Overall, CP Optimizer proved to be the most competitive solver, demonstrating optimality in 38 instances, despite a slightly higher average optimality gap (3.06%). OR-Tools followed closely, achieving a marginally smaller average gap of 2.86%, proving optimality in 23 instances, and consistently producing competitive lower bounds. In third place, Hexaly exhibited improved performance as the instance size increased.***

Keywords: constraint programming, job shop scheduling problem, CP Optimizer, Google OR-Tools, Hexaly.

I. Introduction

Effective production planning lies at the heart of competitive manufacturing and service operations, translating strategic objectives into executable schedules that coordinate people, machines, materials, and information flows. By aligning capacity with demand, production planning reduces lead times, minimizes inventory and work-in-process, and improves on-time delivery, outcomes that directly affect customer satisfaction, cash flow, and profitability [1][2]. Optimization techniques provide the mathematical and algorithmic backbone for these decisions, enabling planners to evaluate trade-offs among conflicting objectives such as makespan, tardiness, throughput, and cost [3]. Rather than relying on ad hoc rules or manual adjustments, optimization embeds operational constraints and business priorities into formal models, producing solutions that are both transparent and reproducible. Scheduling, as the operational layer of production planning, converts optimized plans into concrete sequences of tasks on resources; its quality determines how well theoretical gains translate into shop-floor performance [4]. Good schedules reduce machine idle time, smooth resource utilization, and create slack that absorbs variability, while poor schedules amplify bottlenecks and erode the benefits of upstream planning [5].

Among scheduling problems, the job shop scheduling problem (JSSP) has attracted exceptional attention because it captures the complexity of many real-world production environments: heterogeneous machines, job-specific operation sequences, and tight resource contention [3]. The JSSP is not only a rich theoretical challenge (being NP-hard and a benchmark for combinatorial optimization methods) but also a practical template for industries as diverse as automotive, electronics, aerospace, and pharmaceuticals, where bespoke routing and machine specialization are common. Its ubiquity in practice and its difficulty in theory have made the JSSP a focal point for methodological innovation, driving the development of exact algorithms, heuristics, and hybrid approaches that are then adapted to other scheduling variants. Because many industrial instances remain large, stochastic, or only partially specified, research on the JSSP continues to bridge the gap between idealized models and the messy realities of production systems, producing techniques that scale, adapt, and integrate with enterprise information systems [1][6].

In recent years, constraint programming (CP) has emerged as a leading paradigm for modeling and solving scheduling problems, including the JSSP [7]. CP's declarative modeling style (where constraints express relations among variables rather than procedural steps) matches naturally with scheduling concepts such as precedence, resource capacities, and temporal windows. Advances in propagation algorithms, global constraints (for example, cumulative and no-overlap), and search heuristics have substantially increased CP solvers' ability to prune infeasible regions and find high-quality solutions quickly. This progress, combined with hybridization strategies that integrate CP with mixed-integer programming, machine learning, or metaheuristics, has elevated CP from a niche academic tool to a practical, state-of-the-art technology for scheduling [8] [9] [10]. Commercial and open-source CP solvers now offer robust APIs, parallel search, and sophisticated modeling languages, making it feasible to embed CP into decision support systems and production control loops [11].

The growing ecosystem of CP software presents both opportunities and challenges for practitioners and researchers. Each solver implements different propagation engines and exposes unique tuning features and modeling idioms; some prioritize fast, feasible solutions, while others emphasize strong lower bounds or proof of optimality [12][13]. These architectural and implementation differences can lead to substantial performance variation across instance types, problem sizes, and objective formulations. Consequently, it becomes essential to systematically assess solver capabilities under controlled conditions to understand their strengths, weaknesses, and practical trade-offs. Benchmarking across representative instance sets, standardized formulations, and identical computational environments yields actionable insights for tool selection, model refinement, and hybrid strategy design [11][8].

Given the strategic role of scheduling in achieving organizational goals and the rapid maturation of CP technologies, a careful empirical evaluation of contemporary CP solvers on canonical scheduling benchmarks is timely and necessary (CP Optimizer, Google OR-Tools, and Hexaly) [14][15]. Such an assessment not only informs practitioners about which tools are most effective for particular classes of problems but also guides researchers toward the algorithmic features and modeling constructs that merit further study. By comparing solvers on metrics that matter in practice (solution quality, bound strength, robustness, and runtime behavior), one can better align methodological advances with industrial needs and accelerate the adoption of optimization-driven scheduling in real production environments.

The remainder of the article is organized as follows. In Section II, we provide a brief literature review of the use of CP for scheduling problems, with an emphasis on the JSSP. In particular, we identify the CP software used and any available benchmarks. In Section III, we present the setup of our computational experiments and analyze the results. Finally, Section IV presents the conclusions and outlines avenues for future research.

II. Literature Review and solvers presentation

A. *Job shop scheduling and constraint programming*

The job shop scheduling problem (JSSP) has long served as a central benchmark in scheduling research because it encapsulates the core combinatorial structure of many real-world production systems while posing significant computational challenges. Early work established foundational lower-bounding ideas and constructive procedures (such as time-window shaving and related tightening techniques [16]), while the disjunctive graph representation and tabu search (TS) traditions produced many of the best upper bounds for decades [17]. Subsequent metaheuristic innovations, including Global Equilibrium Search, critical-block neighborhood strategies, backbone-fixing schemes, biased random-key genetic algorithms with local search, path-relinking, and hybrid GA–TS combinations [3] [4], have steadily advanced upper bounds on canonical benchmark sets and demonstrated the practical value of intensification/ diversification strategies for finding high-quality feasible schedules. These heuristic and hybrid approaches remain indispensable for large instances where exact methods are computationally prohibitive.

On the lower-bounding front, research has produced progressively stronger relaxations and inference procedures. Classical analytical bounds (e.g., machine and job load sums) were enhanced by shaving and time-window tightening [16], and more recent translations to SAT and dynamic-programming formulations have yielded competitive bounds for selected instances [18][19]. Constraint programming (CP) has emerged as a particularly influential paradigm for deriving tight bounds: CP's interval variables, global constraints (no overlap, cumulative), and propagation mechanisms enable aggressive pruning of infeasible temporal assignments, which, when combined with systematic search, strengthen lower bounds. Notable CP-based advances include Failure-Directed Search (FDS) and related propagation-centric algorithms that improved many best-known lower bounds on benchmark sets [20][21]. Recent empirical comparisons also indicate that CP formulations often outperform mixed-integer linear programming (MILP) on scheduling benchmarks, both in bound strength and practical solvability [7], reinforcing CP's growing prominence in exact scheduling research.

A parallel and complementary trend is methodological hybridization: researchers increasingly combine CP with MILP relaxations, SAT encodings, metaheuristics, and learning-based components to exploit complementary strengths, CP for expressive modeling and propagation, MILP for tight linear relaxations, and heuristics for rapid feasible solutions. Hybrid schemes have proven effective on large or structured instances where a single paradigm struggles, and they have motivated algorithmic features now embedded in commercial solvers. At the same time, the community has maintained a rich ecosystem of benchmark instances [22], which both stress new methods and preserve long-standing open problems that drive methodological progress.

More recently, automated and learning-based enhancements to CP search (such as reinforcement-learning policies for branching and learned heuristics to reduce search-tree size) have begun to appear [6], aiming to make CP solvers more adaptive across heterogeneous instance families. Commercial and open-source CP platforms (e.g., IBM CP Optimizer, Google OR-Tools, Hexaly) now incorporate many of these advances, yet they differ substantially in propagation engines, supported global constraints, and tuning interfaces; empirical studies show that solver performance is highly instance-dependent, with some engines excelling at quickly finding good feasible solutions and others producing stronger lower bounds or proving optimality more often. This solver heterogeneity underscores the need for systematic benchmarking under controlled conditions.

Two persistent research directions motivate the present study: (i) the search for stronger, scalable lower-bounding procedures that close gaps on long-standing open instances, and (ii) rigorous empirical evaluation of modern CP solvers to guide both practitioners and researchers. Recent CP-based lower-bounding procedures that iteratively tighten Resource-Constrained Project Scheduling Problem (RCPSP) relaxations have demonstrated the potential to improve best-known bounds on open JSSP instances [23][13], illustrating how tailored CP formulations and iterative refinement can yield substantive advances. Building on these strands, the current work benchmarks contemporary CP solvers on canonical JSSP sets and investigates CP-based bounding strategies, thereby contributing empirical evidence and methodological insight to the evolving scheduling literature.

B. CP solvers description

As mentioned, three different CP solvers have been used: CP-Optimizer, OR-Tools, and Hexaly. Next, we introduce the main features and characteristics of each solver to provide a more complete and detailed description.

Firstly, IBM ILOG CP Optimizer is a specialized constraint programming tool designed for complex scheduling and combinatorial problems. It provides a robust "model-and-run" framework that efficiently handles constraints that are difficult to linearize using traditional mathematical programming. Key features include specialized modeling for detailed scheduling via interval variables for activities and cumulative functions for resources. It effectively manages finite-capacity reservoirs, batch setup times, and diverse cost functions such as earliness, tardiness, and non-execution costs. Additionally, it offers specialized constraints, such as all-different, pack, and lexicographic, for business problems involving routing and configuration.

Secondly, Google OR-Tools is an open-source suite for tackling tough challenges in vehicle routing, flows, and integer programming. Its most prominent feature is the award-winning CP-SAT solver, which has consistently won gold in international constraint programming competitions since 2013. OR-Tools is highly portable, supporting development in C++, Python, C#, and Java. It acts as a flexible interface, allowing users to model problems once and then solve them using various engines, including its own open-source solvers, such as GLOP, and third-party commercial solvers, such as Gurobi and CPLEX.

And thirdly, Hexaly offers an innovative approach to optimization through compact modeling based on interval and list variables. This architecture allows for more intuitive and straightforward problem representations than traditional MILP. A standout feature is its use of lambda expressions and varied operators to link sequences of tasks, making it exceptionally powerful for large-scale scheduling. Hexaly is specifically designed for industrial-scale instances, demonstrating the ability to find high-quality solutions to problems involving one million activities within minutes, often outperforming general-purpose solvers at these scales. Like its counterparts, it supports major programming languages including Python, C++, C#, and Java [24].

III. Experimental Setup and Results

A. Experimental Setup

All computational experiments were performed on a computer equipped with an Intel i7-10750H 2.60 GHz 6-core (12-thread) processor and 16 GB of 2933 MHz DDR4 RAM. The latest available versions of the CP solvers at the start of the experiments were used, namely:

- CP Optimizer version 22.1.2
- Google OR-Tools version 9.14.6206
- Hexaly version 14.0

Each software package was configured using its default settings, except for a runtime limit of 600 seconds per instance. All available threads were utilized in the experiments.

The computational implementations were carried out using the official Java APIs of the software packages within the Eclipse IDE for Java Developers (2025-09). In this context, the CP formulation used in each software for the $J_m||C_{max}$ corresponds to the model proposed on the respective developer's website or documentation. For CP Optimizer, this is the formulation provided in the "**SchedJobShop.java**" file located in the installation's examples directory. For Google OR-Tools, we use the Java formulation available on the official documentation website [15]. Finally, for Hexaly, the formulation is taken from the "**Jobshop.java**" file located in the installation's jobshop directory. Due to space limitations, these formulations are not reproduced here.

B. Results

For the computational tests, we used 80 classic JSSP instances from Taillard [22]. These instances have been studied extensively by the scientific community for decades, and to date, many remain *open*, meaning their optimality has not yet been demonstrated [13]. Additionally, for evaluation purposes, we consider the following performance indicators:

$$\mathrm{GAP} = \frac{\mathrm{UB}-\mathrm{LB}}{\mathrm{UB}} \quad (1)$$

$$\mathrm{RPD}_{\mathrm{UB}} = \frac{\mathrm{UB}-\mathrm{Best}_{\mathrm{UB}}}{\mathrm{Best}_{\mathrm{UB}}} \quad (2)$$

$$\mathrm{RPD}_{\mathrm{LB}} = \frac{\mathrm{Best}_{\mathrm{LB}}-\mathrm{LB}}{\mathrm{Best}_{\mathrm{LB}}} \quad (3)$$

Equation (1) defines the optimality gap (GAP), expressed as a percentage, between the upper bound (UB) and the lower bound (LB) reported by the method. Equation (2) presents the relative percentage deviation ($\mathrm{RPD}_{\mathrm{UB}}$) with respect to the best-known upper bound ($\mathrm{Best}_{\mathrm{UB}}$). Finally, Equation (3) presents the $\mathrm{RPD}_{\mathrm{LB}}$ with respect to the best-known lower bound ($\mathrm{Best}_{\mathrm{LB}}$). The $\mathrm{Best}_{\mathrm{LB}}$ and $\mathrm{Best}_{\mathrm{UB}}$ values were taken from Oleg Shylo's web repository ([25]), together with eight improved lower bounds reported recently by Yuraszeck et al. (2026) [13] and highlighted in red in Table II (Taillard's set) of the Appendix.

Table I reports the average values achieved for these indicators by the three software programs across the different combinations of the number of jobs (n) and machines (m). Each combination (row) in the table corresponds to averages computed over 10 instances. Finally, we report the number of instances in which the respective solver successfully demonstrated optimality of the best solution obtained ($\#OPT$), as well as the average execution time, in seconds (t). For ease of comparison, the best results achieved are highlighted in bold.

TABLE I
SUMMARY OF BENCHMARK RESULTS FOR THREE CP SOLVERS ON THE $J_m||C_{max}$ PROBLEM

Instance's Set	n	m	CP Optimizer 22.1.2					OR Tools v9.14.6206					Hexaly 14.0				
			GAP	RPD [UB]	RPD [LB]	# OPT	t	GAP	RPD [UB]	RPD [LB]	# OPT	t	GAP	RPD [UB]	RPD [LB]	# OPT	t
Taillard	15	15	0.78%	0.05%	0.74%	**9**	**138**	**0.31%**	**0.00%**	**0.31%**	**9**	207	8.92%	2.11%	7.02%	0	600
	20		5.66%	1.07%	4.66%	**2**	490	**3.16%**	**0.70%**	**2.47%**	**2**	**486**	7.47%	2.92%	4.77%	0	600
	30		**1.62%**	**1.28%**	0.22%	1	542	2.06%	1.86%	**0.09%**	**2**	**484**	2.97%	2.65%	0.26%	0	600
	50		**0.00%**	**0.00%**	**0.00%**	**10**	**19**	0.06%	0.06%	**0.00%**	9	235	0.06%	0.06%	**0.00%**	9	155
	20	20	9.46%	1.72%	7.13%	0	600	**4.98%**	**0.98%**	**3.22%**	0	600	10.86%	3.27%	7.16%	0	600
	30		**6.79%**	**3.43%**	1.28%	0	600	6.83%	4.16%	**0.63%**	0	600	8.00%	4.75%	1.33%	0	600
	50		**0.15%**	**0.15%**	**0.00%**	**6**	**361**	2.45%	2.54%	**0.00%**	1	555	1.18%	1.21%	**0.00%**	2	573
	100		**0.00%**	**0.00%**	**0.00%**	**10**	194	3.06%	3.17%	**0.00%**	0	600	**0.00%**	**0.00%**	**0.00%**	**10**	**83**
		AVG	3.06%	0.96%	1.75%		368	2.86%	1.68%	0.84%		471	4.93%	2.12%	2.57%		476
		SUM				38					23					21	

Next, we analyze the results obtained:

a) At first glance, the most competitive solvers appear to be CP Optimizer and OR-Tools. In terms of the number of instances for which optimality was demonstrated, CP Optimizer achieved it in 38 instances, followed by OR-Tools in 23 and Hexaly in 21. It is worth noting that the largest differences between CP Optimizer and OR-Tools arise in the 20-machine instances, where CP Optimizer proves optimality in 16 out of 40 instances, compared to only 1 for OR-Tools.

b) Comparatively, Hexaly's performance appears to improve as the instance size increases. This is particularly evident in the 100×20 Taillard. For example, on the 100×20 Taillard instances, Hexaly proved optimality for all instances (as did CP Optimizer), but in less than half the time. In contrast, for this set of instances, OR-Tools fails to demonstrate optimality in any of them.

c) Regarding the average optimality gap, OR-Tools achieves 2.86%, followed closely by CP Optimizer with 3.06%, and then by Hexaly with 4.93%. However, the nearly equivalent performance of CP Optimizer and OR-Tools stems from different factors. Specifically, CP Optimizer attains a lower average RPD_{UB} (0.96%) than OR-Tools (1.68%), whereas OR-Tools outperforms CP Optimizer on average RPD_{LB}, with 0.84% compared to 1.75%. These aggregate results are consistent with the boxplots in Fig. 1, which indicate that CP Optimizer exhibits lower variability in RPD_{UB} compared to OR-Tools, but greater dispersion in RPD_{LB}. Despite this, CP Optimizer achieves a lower median in both performance indicators. Consequently, from a practical perspective, CP Optimizer would be the preferred option, as it tends to produce upper bounds with smaller makespan values.

d) Fig. 1 also presents a histogram of the computation times for each solver. Consistent with the results discussed in a), CP Optimizer exhibits the shortest computation times (as reflected in a histogram that is comparatively left-skewed). In particular, the average computation time for instances in which optimality is demonstrated is approximately 112 seconds, compared to 151 seconds for OR-Tools and 129 seconds for Hexaly.

IV. CONCLUSIONS

In this article, we benchmark three CP software for solving the JSSP with the objective of minimizing the makespan: CP Optimizer, Google OR-Tools, and Hexaly. The software selection criteria considered their popularity within the scheduling community and their free availability for academic use. For evaluation purposes, we used Taillard instances of varying sizes.

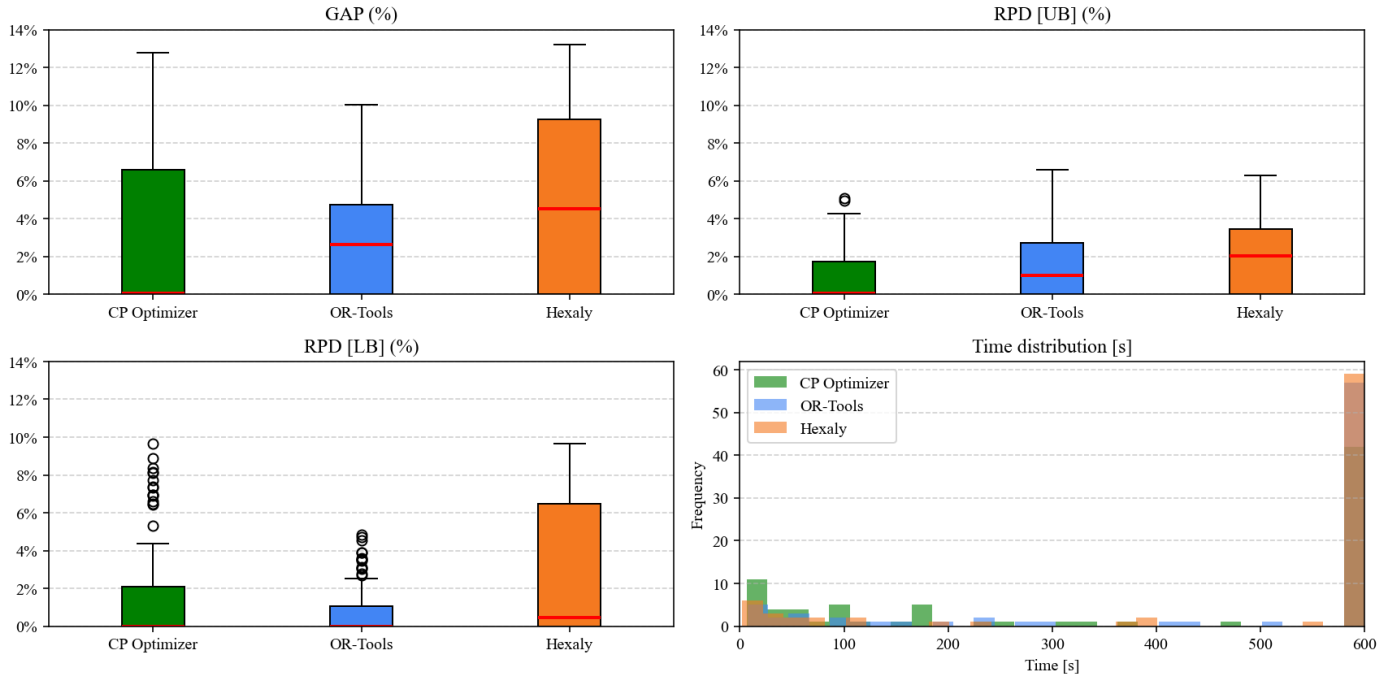

Fig. 1 Boxplots of GAP, RPD_{UB}, and RPD_{LB}, and Histograms of runtime (t) for three CP solvers on the $J_m||C_{max}$ benchmark. For the boxplots, outliers are defined according to Tukey's boxplot rule (1.5·IQR). Interquartile Range (IQR). The median value is indicated by a red line. For the histograms, the maximum runtime was limited to 600 seconds per instance.

The results highlight the competitiveness of CP Optimizer, followed closely by OR-Tools and, in third place, by Hexaly. A notable observation is that OR-Tools consistently produces competitive lower bounds, whereas CP Optimizer is more competitive in terms of upper bounds, emerging as the overall dominant solver.

In light of the results of this work, several avenues for future research can be identified. For instance, the experiments could be extended to larger JSSP instances, where Hexaly is expected to be more competitive (see [24]). Another promising direction is implementing a collaborative approach among the software packages, leveraging their respective strengths. For example, the problem could initially be solved using CP Optimizer, with the search terminated if the upper bound is not improved after a predetermined time, followed by an attempt to strengthen the lower bound using OR-Tools. Additionally, the experimental analysis could be extended to other related scheduling problems, and the inclusion of additional CP software for evaluation could also be considered.

REFERENCES

[1] Pinedo, M. L. (2022). Scheduling: Theory, Algorithms, and Systems, sixth ed. Springer.

[2] Zhang, C., Wu, Y., Ma, Y., Song, W., Le, Z., Cao, Z., & Zhang, J. (2023). A review on learning to solve combinatorial optimisation problems in manufacturing. *IET Collaborative Intelligent Manufacturing*, *5*(1), e12072, doi: 10.1049/cim2.12072.

[3] Dauzère-Pérès, S., Ding, J., Shen, L., & Tamssaouet, K. (2024). The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2), 409-432, doi: 10.1016/j.ejor.2023.05.017.

[4] Xie, J., Li, X., Gao, L., Gui, L., (2022). A hybrid algorithm with a new neighborhood structure for job shop scheduling problems. *Computers & Industrial Engineering*. 169, 108205, doi: 10.1016/j.cie.2022.108205.

[5] Da Col, G., & Teppan, E. C. (2022). Industrial-size job shop scheduling with constraint programming. *Operations Research Perspectives*, *9*, 100249, doi: 10.1016/j.orp.2022.100249.

[6] Heinz, V., Vilím, P., & Hanzálek, Z. (2025). Reinforcement learning for search tree size minimization in constraint programming: New results on scheduling benchmarks. *Computers & Industrial Engineering*, 111413, doi: 10.1016/j.cie.2025.111413.

[7] Naderi, B., Ruiz, R., & Roshanaei, V. (2023). Mixed-integer programming vs. constraint programming for shop scheduling problems: new results and outlook. *INFORMS Journal on Computing*, *35*(4), 817-843, doi: 10.1287/ijoc.2023.1287.

[8] Prata, B. A., Abreu, L. R., & Nagano, M. S. (2024). Applications of constraint programming in production scheduling problems: A descriptive bibliometric analysis. *Results in Control and Optimization*, *14*, 100350, doi: 10.1016/j.rico.2023.100350.

[9] F. Yuraszeck, G. Mejía, D. A. Rossit, and A. Lüer-Villagra. (2025). "A constraint programming-based lower bounding procedure for the job shop scheduling problem," *Computers & Operations Research*, vol. 177, May 2025, Art. no. 106964, doi: 10.1016/j.cor.2024.106964.

[10] Zhang, M., Li, X., Gao, L., & Liu, Q. (2025). Constraint programming-based layered method for integrated process planning and scheduling in extensive flexible manufacturing. *Advanced Engineering Informatics*, *65*, 103210, doi: 10.1016/j.aei.2025.103210.

[11] Müller, D., Müller, M. G., Kress, D., & Pesch, E. (2022). An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. *European Journal of Operational Research*, *302*(3), 874-891, doi: 10.1016/j.ejor.2022.01.034.

[12] Abreu, L. R., & Prata, B. A. (2026). A comparative computational study for single-machine problems with controllable processing times and limited resource consumption. *Annals of Operations Research*, 1-33, doi: 10.1007/s10479-025-07010-y.

[13] Yuraszeck, F., Mejía, G., Rossit, D. A., & Lüer-Villagra, A. (2026). A note on "A constraint programming-based lower bounding procedure for the job shop scheduling problem". *Computers & Operations Research*, vol. 189, May 2026, Art. no. 107396, doi: 10.1016/j.cor.2026.107396.

[14] Hexaly, "Hexaly vs. CP Optimizer vs. OR-Tools on the Job Shop Scheduling Problem (JSSP)," Hexaly Benchmarks, [Online]. Available: https://www.hexaly.com/benchmarks/hexaly-vs-cp-optimizer-vs-or-tools-on-the-job-shop-scheduling-problem-jssp. [Accessed: Jan. 5, 2026].

[15] Google, "The Job Shop Problem," Google Optimization Tools (OR-Tools), [Online]. Available: https://developers.google.com/optimization/scheduling/job_shop. [Accessed: Jan. 5, 2026].

[16] Jain, A., Meeran, S., (1999). Theory and methodology deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research,* doi: 10.1016/S0377-2217(98)00113-1.

[17] Nowicki, E., Smutnicki, C., (2005). An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*. 8 (2), 145–159, doi: 10.1007/s10951-005-6364-5.

[18] Koshimura, M., Nabeshima, H., Fujita, H., Hasegawa, R., (2010). Solving open job-shop scheduling problems by SAT encoding. In: *IEICE Transactions on Information and Systems*. vol. E93-D, Institute of Electronics, Information and Communication, Engineers, IEICE, pp. 2316–2318.

[19] Hoorn, J.V., (2016). Dynamic programming for routing and scheduling: optimizing sequences of decisions. (Ph.D. thesis). Vrije Universiteit Amsterdam.

[20] Vilím, P., Laborie, P., Shaw, P., (2015). Failure-directed search for constraint-based scheduling. In: *Integration of AI and OR Techniques in Constraint Programming:12th International Conference, CPAIOR 2015*, Barcelona, Spain, May 18-22, 2015, Proceedings 12. Springer, pp. 437–453, doi: 10.1007/978-3-319-18008-3_30.

[21] Siala, M., Artigues, C., Hebrard, E., (2015). Two clause learning approaches for disjunctive scheduling. In: *International Conference on Principles and Practice of Constraint Programming*. Springer, pp. 393–402, doi: 10.1007/978-3-319-23219-5_28.

[22] Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, *64*(2), 278-285, doi: 10.1016/0377-2217(93)90182-M.

[23] Yuraszeck, F., Mejía, G., & Lüer-Villagra, A. (2025). An adapted constraint-programming formulation of the resource-constrained project scheduling problem applied to the identical parallel machines group shop and mixed shop scheduling problems. *International Transactions in Operational Research*, *32*(3), 1422-1441, doi: 10.1111/itor.13504.

[24] Blaise, L. (2025). Modeling scheduling problems with Hexaly. In *11th IFAC Conference on Manufacturing Modelling, Management and Control–IFAC MIM2025; Trondheim, Norway*.

[25] Shylo, O. V., & Shams, H. (2018). Boosting binary optimization via binary classification: A case study of job shop scheduling. *arXiv preprint arXiv:1808.10813*.

APPENDIX A

TABLE II
DETAILS OF BENCHMARK RESULTS FOR THREE CP SOLVERS ON THE $J_m||C_{max}$ PROBLEM FOR TAILLARD'S INSTANCES

Inst.	*n*	*m*	Best LB	Best UB	GAP	CP Optimizer 22.1.2						OR Tools v9.14.6206						Hexaly 14.0					
						LB	UB	GAP	RPD [UB]	RPD [LB]	*t*	LB	UB	GAP	RPD [UB]	RPD [LB]	*t*	LB	UB	GAP	RPD [UB]	RPD [LB]	*t*
TA01	15	15	**1231**	**1231**	**0.00%**	**1231**	**1231**	**0.00%**	0.00%	0.00%	14.31	**1231**	**1231**	**0.00%**	0.00%	0.00%	7.22	1190	1258	5.41%	2.19%	3.33%	600
TA02	15	15	**1244**	**1244**	**0.00%**	**1244**	**1244**	**0.00%**	0.00%	0.00%	50.18	**1244**	**1244**	**0.00%**	0.00%	0.00%	93.47	1167	1247	6.42%	0.24%	6.19%	600
TA03	15	15	**1218**	**1218**	**0.00%**	**1218**	**1218**	**0.00%**	0.00%	0.00%	23.72	**1218**	**1218**	**0.00%**	0.00%	0.00%	26.07	1137	1234	7.86%	1.31%	6.65%	600
TA04	15	15	**1175**	**1175**	**0.00%**	**1175**	**1175**	**0.00%**	0.00%	0.00%	37.67	**1175**	**1175**	**0.00%**	0.00%	0.00%	17.01	1097	1207	9.11%	2.72%	6.64%	600
TA05	15	15	**1224**	**1224**	**0.00%**	**1224**	**1224**	**0.00%**	0.00%	0.00%	179.29	**1224**	**1224**	**0.00%**	0.00%	0.00%	232.33	1120	1271	11.88%	3.84%	8.50%	600
TA06	15	15	**1238**	**1238**	**0.00%**	1147	1244	7.80%	0.48%	7.35%	600	1200	1238	3.07%	0.00%	3.07%	600	1147	1265	9.33%	2.18%	7.35%	600
TA07	15	15	**1227**	**1227**	**0.00%**	**1227**	**1227**	**0.00%**	0.00%	0.00%	156.77	**1227**	**1227**	**0.00%**	0.00%	0.00%	405.39	1146	1245	7.95%	1.47%	6.60%	600
TA08	15	15	**1217**	**1217**	**0.00%**	**1217**	**1217**	**0.00%**	0.00%	0.00%	173.41	**1217**	**1217**	**0.00%**	0.00%	0.00%	144.05	1118	1240	9.84%	1.89%	8.13%	600
TA09	15	15	**1274**	**1274**	**0.00%**	**1274**	**1274**	**0.00%**	0.00%	0.00%	92.79	**1274**	**1274**	**0.00%**	0.00%	0.00%	513.56	1157	1305	11.34%	2.43%	9.18%	600
TA10	15	15	**1241**	**1241**	**0.00%**	**1241**	**1241**	**0.00%**	0.00%	0.00%	47.29	**1241**	**1241**	**0.00%**	0.00%	0.00%	28.94	1147	1276	10.11%	2.82%	7.57%	600
TA11	20	15	**1357**	**1357**	**0.00%**	1269	1383	8.24%	1.92%	6.48%	600	1309	1374	4.73%	1.25%	3.54%	600	1269	1410	10.00%	3.91%	6.48%	600
TA12	20	15	**1367**	**1367**	**0.00%**	1314	1373	4.30%	0.44%	3.88%	600	1350	1367	1.24%	0.00%	1.24%	600	1314	1401	6.21%	2.49%	3.88%	600
TA13	20	15	**1342**	**1342**	**0.00%**	1243	1355	8.27%	0.97%	7.38%	600	1277	1352	5.55%	0.75%	4.84%	600	1243	1378	9.80%	2.68%	7.38%	600
TA14	20	15	**1345**	**1345**	**0.00%**	**1345**	**1345**	**0.00%**	0.00%	0.00%	10.33	**1345**	**1345**	**0.00%**	0.00%	0.00%	13.34	1345	1355	0.74%	0.74%	0.00%	600
TA15	20	15	**1339**	**1339**	**0.00%**	1246	1372	9.18%	2.46%	6.95%	600	1302	1352	3.70%	0.97%	2.76%	600	1246	1374	9.32%	2.61%	6.95%	600
TA16	20	15	**1360**	**1360**	**0.00%**	1250	1374	9.02%	1.03%	8.09%	600	1296	1369	5.33%	0.66%	4.71%	600	1250	1401	10.78%	3.01%	8.09%	600
TA17	20	15	**1462**	**1462**	**0.00%**	**1462**	**1462**	**0.00%**	0.00%	0.00%	86.24	**1462**	**1462**	**0.00%**	0.00%	0.00%	51.34	1445	1509	4.24%	3.21%	1.16%	600
TA18	20	15	1393	1396	0.21%	1358	1414	3.96%	1.29%	2.51%	600	1360	1406	3.27%	0.72%	2.37%	600	1358	1434	5.30%	2.72%	2.51%	600
TA19	20	15	**1332**	**1332**	**0.00%**	1240	1356	8.55%	1.80%	6.91%	600	1296	1357	4.50%	1.88%	2.70%	600	1240	1395	11.11%	4.73%	6.91%	600
TA20	20	15	**1348**	**1348**	**0.00%**	1289	1358	5.08%	0.74%	4.38%	600	1314	1359	3.31%	0.82%	2.52%	600	1289	1389	7.20%	3.04%	4.38%	600
TA21	20	20	**1642**	**1642**	**0.00%**	1508	1668	9.59%	1.58%	8.16%	600	1578	1654	4.59%	0.73%	3.90%	600	1508	1681	10.29%	2.38%	8.16%	600
TA22	20	20	1585	1600	0.94%	1444	1651	12.54%	3.19%	8.90%	600	1528	1616	5.45%	1.00%	3.60%	600	1441	1655	12.93%	3.44%	9.09%	600
TA23	20	20	1541	1557	1.03%	1439	1591	9.55%	2.18%	6.62%	600	1493	1567	4.72%	0.64%	3.11%	600	1439	1581	8.98%	1.54%	6.62%	600
TA24	20	20	**1644**	**1644**	**0.00%**	1538	1649	6.73%	0.30%	6.45%	600	1610	1650	2.42%	0.36%	2.07%	600	1538	1695	9.26%	3.10%	6.45%	600
TA25	20	20	1584	1595	0.69%	1474	1604	8.10%	0.56%	6.94%	600	1529	1611	5.09%	1.00%	3.47%	600	1473	1662	11.37%	4.20%	7.01%	600
TA26	20	20	1615	1645	1.82%	1490	1694	12.04%	2.98%	7.74%	600	1558	1678	7.15%	2.01%	3.53%	600	1490	1705	12.61%	3.65%	7.74%	600
TA27	20	20	1673	1680	0.42%	1584	1709	7.31%	1.73%	5.32%	600	1622	1706	4.92%	1.55%	3.05%	600	1584	1786	11.31%	6.31%	5.32%	600
TA28	20	20	**1603**	**1603**	**0.00%**	1552	1619	4.14%	1.00%	3.18%	600	1587	1620	2.04%	1.06%	1.00%	600	1552	1658	6.39%	3.43%	3.18%	600
TA29	20	20	1606	1625	1.17%	1451	1646	11.85%	1.29%	9.65%	600	1533	1639	6.47%	0.86%	4.55%	600	1451	1654	12.27%	1.78%	9.65%	600
TA30	20	20	1543	1584	2.59%	1414	1621	12.77%	2.34%	8.36%	600	1483	1594	6.96%	0.63%	3.89%	600	1414	1629	13.20%	2.84%	8.36%	600
TA31	30	15	**1764**	**1764**	**0.00%**	1764	1766	0.11%	0.11%	0.00%	600	1764	1766	0.11%	0.11%	0.00%	600	1764	1790	1.45%	1.47%	0.00%	600
TA32	30	15	1774	1784	0.56%	1774	1834	3.27%	2.80%	0.00%	600	1774	1860	4.62%	4.26%	0.00%	600	1774	1864	4.83%	4.48%	0.00%	600
TA33	30	15	1790	1791	0.06%	1774	1840	3.59%	2.74%	0.89%	600	1783	1834	2.78%	2.40%	0.39%	600	1774	1881	5.69%	5.03%	0.89%	600
TA34	30	15	**1828**	**1828**	**0.00%**	1828	1863	1.88%	1.91%	0.00%	600	1828	1878	2.66%	2.74%	0.00%	600	1828	1880	2.77%	2.84%	0.00%	600
TA35	30	15	**2007**	**2007**	**0.00%**	**2007**	**2007**	**0.00%**	0.00%	0.00%	23.55	**2007**	**2007**	**0.00%**	0.00%	0.00%	10.07	1999	2015	0.79%	0.40%	0.40%	600
TA36	30	15	**1819**	**1819**	**0.00%**	1819	1828	0.49%	0.49%	0.00%	600	1819	1831	0.66%	0.66%	0.00%	600	1819	1837	0.98%	0.99%	0.00%	600
TA37	30	15	**1771**	**1771**	**0.00%**	1771	1784	0.73%	0.73%	0.00%	600	1771	1823	2.85%	2.94%	0.00%	600	1771	1835	3.49%	3.61%	0.00%	600
TA38	30	15	**1673**	**1673**	**0.00%**	1673	1685	0.71%	0.72%	0.00%	600	1673	1693	1.18%	1.20%	0.00%	600	1673	1731	3.35%	3.47%	0.00%	600
TA39	30	15	**1795**	**1795**	**0.00%**	1795	1806	0.61%	0.61%	0.00%	600	**1795**	**1795**	**0.00%**	0.00%	0.00%	31.83	1795	1818	1.27%	1.28%	0.00%	600
TA40	30	15	1652	1670	1.08%	1631	1714	4.84%	2.63%	1.27%	600	1643	1742	5.68%	4.31%	0.54%	600	1631	1719	5.12%	2.93%	1.27%	600
TA41	30	20	1912	2006	4.69%	1857	2087	11.02%	4.04%	2.88%	600	1890	2074	8.87%	3.39%	1.15%	600	1854	2121	12.59%	5.73%	3.03%	600
TA42	30	20	1887	1939	2.68%	1867	1998	6.56%	3.04%	1.06%	600	1874	1998	6.21%	3.04%	0.69%	600	1867	2006	6.93%	3.46%	1.06%	600
TA43	30	20	1809	1846	2.00%	1809	1892	4.39%	2.49%	0.00%	600	1809	1947	7.09%	5.47%	0.00%	600	1809	1960	7.70%	6.18%	0.00%	600
TA44	30	20	1952	1979	1.36%	1923	2033	5.41%	2.73%	1.49%	600	1937	2092	7.41%	5.71%	0.77%	600	1923	2079	7.50%	5.05%	1.49%	600
TA45	30	20	1997	2000	0.15%	1990	2034	2.16%	1.70%	0.35%	600	1997	2037	1.96%	1.85%	0.00%	600	1984	2038	2.65%	1.90%	0.65%	600
TA46	30	20	1966	2006	1.99%	1940	2108	7.97%	5.08%	1.32%	600	1943	2079	6.54%	3.64%	1.17%	600	1940	2096	7.44%	4.49%	1.32%	600
TA47	30	20	1816	1889	3.86%	1781	1948	8.57%	3.12%	1.93%	600	1798	1997	9.96%	5.72%	0.99%	600	1781	1989	10.46%	5.29%	1.93%	600
TA48	30	20	1915	1937	1.14%	1905	1992	4.37%	2.84%	0.52%	600	1912	2032	5.91%	4.90%	0.16%	600	1905	2028	6.07%	4.70%	0.52%	600
TA49	30	20	1934	1960	1.33%	1903	2044	6.90%	4.29%	1.60%	600	1926	2012	4.27%	2.65%	0.41%	600	1903	2049	7.13%	4.54%	1.60%	600
TA50	30	20	1837	1923	4.47%	1806	2018	10.51%	4.94%	1.69%	600	1820	2023	10.03%	5.20%	0.93%	600	1806	2041	11.51%	6.14%	1.69%	600
TA51	50	15	**2760**	**2760**	**0.00%**	**2760**	**2760**	**0.00%**	0.00%	0.00%	25.39	**2760**	**2760**	**0.00%**	0.00%	0.00%	267.08	**2760**	**2760**	**0.00%**	0.00%	0.00%	64
TA52	50	15	**2756**	**2756**	**0.00%**	**2756**	**2756**	**0.00%**	0.00%	0.00%	19.75	**2756**	**2756**	**0.00%**	0.00%	0.00%	297.55	**2756**	**2756**	**0.00%**	0.00%	0.00%	235
TA53	50	15	**2717**	**2717**	**0.00%**	**2717**	**2717**	**0.00%**	0.00%	0.00%	14.47	**2717**	**2717**	**0.00%**	0.00%	0.00%	108.91	**2717**	**2717**	**0.00%**	0.00%	0.00%	42
TA54	50	15	**2839**	**2839**	**0.00%**	**2839**	**2839**	**0.00%**	0.00%	0.00%	6.73	**2839**	**2839**	**0.00%**	0.00%	0.00%	62.08	**2839**	**2839**	**0.00%**	0.00%	0.00%	4
TA55	50	15	**2679**	**2679**	**0.00%**	**2679**	**2679**	**0.00%**	0.00%	0.00%	30.24	2679	2695	0.59%	0.60%	0.00%	600	2679	2696	0.63%	0.63%	0.00%	600
TA56	50	15	**2781**	**2781**	**0.00%**	**2781**	**2781**	**0.00%**	0.00%	0.00%	11.89	**2781**	**2781**	**0.00%**	0.00%	0.00%	427.44	**2781**	**2781**	**0.00%**	0.00%	0.00%	35
TA57	50	15	**2943**	**2943**	**0.00%**	**2943**	**2943**	**0.00%**	0.00%	0.00%	11.93	**2943**	**2943**	**0.00%**	0.00%	0.00%	92.26	**2943**	**2943**	**0.00%**	0.00%	0.00%	42
TA58	50	15	**2885**	**2885**	**0.00%**	**2885**	**2885**	**0.00%**	0.00%	0.00%	9.58	**2885**	**2885**	**0.00%**	0.00%	0.00%	196.01	**2885**	**2885**	**0.00%**	0.00%	0.00%	8
TA59	50	15	**2655**	**2655**	**0.00%**	**2655**	**2655**	**0.00%**	0.00%	0.00%	32.08	**2655**	**2655**	**0.00%**	0.00%	0.00%	232.96	**2655**	**2655**	**0.00%**	0.00%	0.00%	398
TA60	50	15	**2723**	**2723**	**0.00%**	**2723**	**2723**	**0.00%**	0.00%	0.00%	27.94	**2723**	**2723**	**0.00%**	0.00%	0.00%	61.63	**2723**	**2723**	**0.00%**	0.00%	0.00%	120
TA61	50	20	**2868**	**2868**	**0.00%**	**2868**	**2868**	**0.00%**	0.00%	0.00%	65.16	2868	2935	2.28%	2.34%	0.00%	600	2868	2882	0.49%	0.49%	0.00%	600
TA62	50	20	**2869**	**2869**	**0.00%**	2869	2900	1.07%	1.08%	0.00%	600	2869	3032	5.38%	5.68%	0.00%	600	2869	2960	3.07%	3.17%	0.00%	600
TA63	50	20	**2755**	**2755**	**0.00%**	2755	2764	0.33%	0.33%	0.00%	600	2755	2841	3.03%	3.12%	0.00%	600	2755	2785	1.08%	1.09%	0.00%	600
TA64	50	20	**2702**	**2702**	**0.00%**	**2702**	**2702**	**0.00%**	0.00%	0.00%	122.54	2702	2794	3.29%	3.40%	0.00%	600	**2702**	**2702**	**0.00%**	0.00%	0.00%	393
TA65	50	20	**2725**	**2725**	**0.00%**	**2725**	**2725**	**0.00%**	0.00%	0.00%	480.98	2725	2839	4.02%	4.18%	0.00%	600	2725	2776	1.84%	1.87%	0.00%	600
TA66	50	20	**2845**	**2845**	**0.00%**	2845	2846	0.04%	0.04%	0.00%	600	2845	2868	0.80%	0.81%	0.00%	600	2845	2884	1.35%	1.37%	0.00%	600
TA67	50	20	**2825**	**2825**	**0.00%**	2825	2826	0.04%	0.04%	0.00%	600	2825	2874	1.70%	1.73%	0.00%	600	2825	2856	1.09%	1.10%	0.00%	600
TA68	50	20	**2784**	**2784**	**0.00%**	**2784**	**2784**	**0.00%**	0.00%	0.00%	63.11	2784	2845	2.14%	2.19%	0.00%	600	2784	2786	0.07%	0.07%	0.00%	600
TA69	50	20	**3071**	**3071**	**0.00%**	**3071**	**3071**	**0.00%**	0.00%	0.00%	99.32	**3071**	**3071**	**0.00%**	0.00%	0.00%	150.27	**3071**	**3071**	**0.00%**	0.00%	0.00%	541
TA70	50	20	**2995**	**2995**	**0.00%**	**2995**	**2995**	**0.00%**	0.00%	0.00%	378.04	2995	3052	1.87%	1.90%	0.00%	600	2995	3083	2.85%	2.94%	0.00%	600
TA71	100	20	**5464**	**5464**	**0.00%**	**5464**	**5464**	**0.00%**	0.00%	0.00%	321.66	5464	5713	4.36%	4.56%	0.00%	600	**5464**	**5464**	**0.00%**	0.00%	0.00%	8
TA72	100	20	**5181**	**5181**	**0.00%**	**5181**	**5181**	**0.00%**	0.00%	0.00%	172.55	5181	5352	3.20%	3.30%	0.00%	600	**5181**	**5181**	**0.00%**	0.00%	0.00%	22
TA73	100	20	**5568**	**5568**	**0.00%**	**5568**	**5568**	**0.00%**	0.00%	0.00%	248.42	5568	5721	2.67%	2.75%	0.00%	600	**5568**	**5568**	**0.00%**	0.00%	0.00%	70
TA74	100	20	**5339**	**5339**	**0.00%**	**5339**	**5339**	**0.00%**	0.00%	0.00%	89.24	5339	5429	1.66%	1.69%	0.00%	600	**5339**	**5339**	**0.00%**	0.00%	0.00%	22
TA75	100	20	**5392**	**5392**	**0.00%**	**5392**	**5392**	**0.00%**	0.00%	0.00%	240.64	5392	5748	6.19%	6.60%	0.00%	600	**5392**	**5392**	**0.00%**	0.00%	0.00%	196
TA76	100	20	**5342**	**5342**	**0.00%**	**5342**	**5342**	**0.00%**	0.00%	0.00%	178.16	5342	5484	2.59%	2.66%	0.00%	600	**5342**	**5342**	**0.00%**	0.00%	0.00%	11
TA77	100	20	**5436**	**5436**	**0.00%**	**5436**	**5436**	**0.00%**	0.00%	0.00%	165.86	5436	5531	1.72%	1.75%	0.00%	600	**5436**	**5436**	**0.00%**	0.00%	0.00%	118
TA78	100	20	**5394**	**5394**	**0.00%**	**5394**	**5394**	**0.00%**	0.00%	0.00%	84.88	5394	5532	2.49%	2.56%	0.00%	600	**5394**	**5394**	**0.00%**	0.00%	0.00%	16
TA79	100	20	**5358**	**5358**	**0.00%**	**5358**	**5358**	**0.00%**	0.00%	0.00%	101.11	5358	5505	2.67%	2.74%	0.00%	600	**5358**	**5358**	**0.00%**	0.00%	0.00%	2
TA80	100	20	**5183**	**5183**	**0.00%**	**5183**	**5183**	**0.00%**	0.00%	0.00%	340.87	5183	5345	3.03%	3.13%	0.00%	600	**5183**	**5183**	**0.00%**	0.00%	0.00%	361