Job shop scheduling with FIFO constraints: A comparative study of MILP and constraint programming

Randa Ouchene[1], Djamal Rebaine[1], Pierre Baptiste[2]

***Abstract*— We consider the Job Shop Scheduling Problem with First-In-First-Out constraints (FIFO-JSSP) and the objective of minimizing the makespan. Enforcing FIFO introduces strong inter-machine coupling, which increases the difficulty of exact optimization. We compare a Mixed-Integer Linear Programming (MILP) baseline with two Constraint Programming (CP) approaches: a direct CP model enforcing FIFO on all machines and a two-phase CP strategy that first enforces FIFO on an instance-dependent strategic machine to obtain a guiding schedule, then restores FIFO on all machines while exploiting warm-start information. On standard benchmark instances solved to optimality by all approaches within the time limit, direct CP and the two-phase strategy are, on average, 3.80 times and 6.24 times faster than MILP, respectively, based on arithmetic mean runtimes. On diversified generated instances, the two-phase approach is 141 times faster than direct CP on bottleneck-centered instances and 1.70 times faster on balanced-workload instances, confirming that the benefit strongly depends on instance structure.**

I. Introduction and related literature

The Job Shop Scheduling Problem (JSP) is one of the most widely studied combinatorial optimization problems in scheduling. The classical JSP and several of its variants are NP-hard, making it a long-standing benchmark for the development of exact and approximate solution methods [6], [18].

The literature on the JSP is traditionally structured around exact methods and heuristic approaches. Exact approaches include branch-and-bound and branch-and-cut algorithms, dynamic programming techniques for restricted cases, and mathematical programming formulations solved with Mixed-Integer Linear Programming (MILP) solvers [10], [4]. Among MILP formulations, the disjunctive model has emerged as a reference approach for makespan minimization and is often reported as one of the most effective exact formulations for the JSP [17].

In parallel, extensive research has investigated heuristic approaches, including constructive heuristics, dispatching rules, local search procedures, and metaheuristics. These approaches typically sacrifice optimality guarantees in exchange for scalability and fast solution times [15].

Over the last decade, Constraint Programming (CP) has been intensively re-evaluated and has emerged as a powerful paradigm for solving hard scheduling problems. This evolution is largely driven by advances in propagation algorithms, the development of dedicated global constraints for scheduling, and the significant performance improvements of modern CP solvers [8]. Several studies report competitive or superior performance of CP compared to MILP on classical JSP instances [11], [7].

Beyond manufacturing-oriented settings, JSP variants are also motivated by service and human-centered contexts, where the order of service is not merely an operational constraint but a fairness requirement. First-Come-First-Served (FIFO) is a canonical discipline in queueing and service systems and is widely perceived as equitable by customers and operators. Its fairness properties have been analyzed from multiple perspectives, including ethical justifications of waiting-time rules, quantitative fairness measures in queues, and axiomatic approaches to fair service disciplines [16].

Despite this extensive literature on queueing, the integration of FIFO ordering constraints into combinatorial JSP optimization models remains limited [12]. In particular, there is a lack of studies addressing how FIFO constraints can be efficiently handled within CP frameworks, especially when compared to the rich literature on no-wait and blocking constraints in shop scheduling [2]. When FIFO is considered, little guidance is available on search strategies that explicitly exploit both bottleneck structures and FIFO-induced coupling between machines. This paper aims to fill this gap by proposing efficient CP approaches for the FIFO-JSP and by providing a comparative analysis with MILP formulations. We study two CP strategies: (i) a direct CP model enforcing FIFO constraints globally on all machines, and (ii) a two-phase CP strategy designed to reduce search effort. The two-phase strategy first activates FIFO only on a strategic machine selected using predefined rules, and then restores FIFO constraints on all machines using warm-start information together with progressively tightened makespan bounds.

The remainder of the paper is organized as follows. Section 2 introduces the problem definition and notation. Section 3 presents the models and solution approaches (MILP, direct CP, and the proposed two-phase CP method). Section 4 details the computational methodology and experimental setup. Section 5 reports and analyzes the experimental results. Section 6 discusses the main findings and their implications. Finally, Section 7 concludes the paper and outlines directions for future work.

[1]Département d'informatique et de mathématique, Université du Québec à Chicoutimi (UQAC), Saguenay (Québec), Canada. randa.ouchene1@uqac.ca, drebaine@uqac.ca

[2]Département de mathématiques et de génie industriel, École Polytechnique Montréal, Montréal (Québec), Canada. pierre.baptiste@polymtl.ca

II. Problem definition and notation

We consider a set of jobs processed on a set of machines. Each job follows a predetermined technological route composed of non-preemptive operations. The classical JSP includes (i) precedence constraints along each job route and (ii) capacity constraints on machines (at most one operation at a time). In what follows, we introduce the notation used throughout the paper and formally define the FIFO constraint in the job shop context.

A. Notation and parameters

We consider n jobs and m machines.

a) Sets and indices:

- $J = \{1, \dots, n\}$: set of jobs, indexed by i.
- $M = \{1, \dots, m\}$: set of machines, indexed by j.
- $k \in \{1, \dots, m\}$: position of an operation in the route of a job.

b) Parameters:

- $p_{ij} > 0$: processing time of job i on machine j.
- $o_{i,k} \in M$: machine processed at position k in the route of job i.
- $\mathrm{pred}_{ij} \in \{-1\} \cup M$: immediate predecessor machine of machine j in the route of job i; $\mathrm{pred}_{ij} = -1$ if j is the first machine visited by job i.

c) Decision variables:

- op_{ij}: interval decision variable representing the operation of job i on machine j (fixed duration p_{ij}), with start time $\mathrm{start}(op_{ij})$ and completion time $\mathrm{end}(op_{ij})$.

B. FIFO ordering rule

In this work, the FIFO rule is enforced as follows: if a job completes earlier on the machine that immediately precedes a common next machine, then it must also be executed earlier on that common next machine.

Let J_1 and J_2 be two distinct jobs. Let $k_1, k_2, h \in M$ be machines such that: (i) in the route of J_1, machine k_1 is immediately followed by machine h, (ii) in the route of J_2, machine k_2 is immediately followed by machine h, (iii) h is the next immediate machine for both jobs in their respective routes.

Denote by C_{J_1,k_1} and C_{J_2,k_2} and the completion times on the preceding machines, and by $S_{J_1,h}$ and $S_{J_2,h}$ the start times on the common next machine h. The FIFO rule is formalized by the implication:

$$C_{J_1,k_1} \leq C_{J_2,k_2} \;\Rightarrow\; S_{J_1,h} \leq S_{J_2,h}.$$

Equivalently, if $C_{J_1,k_1} \leq C_{J_2,k_2}$, then job J_1 must be scheduled before job J_2 on the common next machine h. FIFO is imposed only when the two jobs enter the same next-machine queue. If their next machines are different, no FIFO relation is imposed at that stage.

III. Models and Solution Approaches

This section presents the three exact approaches compared in this paper: the MILP baseline, the direct CP model, and the proposed two-phase CP strategy.

A. MILP formulation

As an exact reference approach, we use a disjunctive MILP formulation for the FIFO-JSSP. Let S_{ij} denote the start time of the operation of job i on machine j, and let $C_{\max}$ denote the makespan. For each pair of jobs $i < i'$ processed on the same machine j, let $x_{ii'j}$ be a binary variable equal to 1 if job i is processed before job i' on machine j, and 0 otherwise. Let H be a sufficiently large constant.

The objective is

$$\min C_{\max}.$$

Technological precedence constraints are given by

$$S_{i,o_{i,k+1}} \geq S_{i,o_{i,k}} + p_{i,o_{i,k}}, \qquad i \in J,\ k = 1, \dots, m-1.$$

Machine-capacity constraints are modeled by the following disjunctive constraints:

$$S_{i'j} \geq S_{ij} + p_{ij} - H(1 - x_{ii'j}), \qquad i < i',\ j \in M,$$

$$S_{ij} \geq S_{i'j} + p_{i'j} - H x_{ii'j}, \qquad i < i',\ j \in M.$$

The makespan is defined by

$$C_{\max} \geq S_{ij} + p_{ij}, \qquad i \in J,\ j \in M.$$

FIFO constraints link the order on a machine j to the order in which jobs become available from their predecessor machines. For every pair $i < i'$ such that $\mathrm{pred}_{ij} \neq -1$ and $\mathrm{pred}_{i'j} \neq -1$, we impose

$$S_{i,\mathrm{pred}_{ij}} + p_{i,\mathrm{pred}_{ij}} \leq S_{i',\mathrm{pred}_{i'j}} + p_{i',\mathrm{pred}_{i'j}} + H(1 - x_{ii'j}),$$

$$S_{i',\mathrm{pred}_{i'j}} + p_{i',\mathrm{pred}_{i'j}} \leq S_{i,\mathrm{pred}_{ij}} + p_{i,\mathrm{pred}_{ij}} + H x_{ii'j}.$$

This MILP extends the classical disjunctive JSP formulation by enforcing FIFO-consistent ordering decisions and is used as the exact baseline in our computational study.

B. Direct CP model

The direct CP model is derived from a classical CP formulation for the JSP based on interval variables and global scheduling constraints (see, e.g., [11]). For each operation of job i on machine j, we define an interval variable op_{ij} with fixed duration p_{ij}. Technological precedence constraints are enforced by

$$\mathrm{end}(\mathrm{op}_{i,o_{i,k}}) \leq \mathrm{start}(\mathrm{op}_{i,o_{i,k+1}}), \qquad i \in J,\ k = 1, \dots, m-1.$$

Machine-capacity constraints are imposed using a no-overlap constraint on the operations assigned to each machine.

FIFO constraints: FIFO is enforced through the following rule: consider a machine $j \in M$ and two jobs $i < i'$ that both visit j. Let $\mathrm{pred}_{ij} \neq -1$ and $\mathrm{pred}_{i'j} \neq -1$ be the immediate predecessor machines of j in the routes of jobs i and i'. FIFO requires that the order on the next common machine j is consistent with the order induced by completion times on the predecessor machines. This is captured by

$$\left(\mathrm{end}(\mathrm{op}_{i,\mathrm{pred}_{ij}}) < \mathrm{end}(\mathrm{op}_{i',\mathrm{pred}_{i'j}})\right) \iff \left(\mathrm{start}(\mathrm{op}_{ij}) \leq \mathrm{start}(\mathrm{op}_{i'j})\right)$$

for all machines $j \in M$ and all pairs of jobs $i < i'$ such that $\mathrm{pred}_{ij} \neq -1$ and $\mathrm{pred}_{i'j} \neq -1$.

The direct CP approach enforces these FIFO constraints on all machines from the beginning of the search and minimizes the makespan under the fixed time limit.

C. Two-phase CP strategy

The two-phase approach reduces early coupling by first enforcing FIFO on a single strategic machine, then activating FIFO on all machines.

a) Phase 1 (partial FIFO): FIFO constraints are imposed only on a selected machine $j^\star$.

Strategic machine selection: The selection combines workload and route centrality. Let

$$\mathrm{load}(j) = \sum_{i \in J} p_{ij}$$

be the workload of machine j. Let

$$\mathrm{pos}(j) = \frac{1}{n} \sum_{i \in J} \sum_{k=1}^{m} k \cdot \mathbb{1}[o_{i,k} = j]$$

be the average position of machine j in the job routes, and let $\mathrm{target} = (m+1)/2$ denote the central position. Define

$$\mathrm{cent}(j) = \frac{1}{1 + |\mathrm{pos}(j) - \mathrm{target}|}, \qquad \mathrm{score}(j) = \mathrm{load}(j) \cdot \mathrm{cent}(j).$$

The strategic machine is chosen as

$$j^\star \in \arg\max_{j \in M} \ \mathrm{score}(j).$$

b) Phase 2 (full FIFO with injection): FIFO constraints are then activated on all machines. The best solution obtained in Phase 1 is used as a warm start to initialize the search.

IV. Computational methodology

A. Computing environment

All experiments were conducted with IBM ILOG CPLEX Optimization Studio 20.1 on a computing cluster composed of 11 Dell PowerEdge R740 servers. Each node is equipped with 512 GB RAM and Intel Xeon Gold 6258R processors running at 2.70 GHz.

B. Test instances

To assess the performance of the proposed approaches, we used well-established benchmark instances from the literature [1], [3], [5], [9], [14], along with 100 newly generated synthetic instances designed to evaluate and contrast the two CP strategies under controlled structural features.

C. Compared approaches and settings

We evaluate three exact approaches: (i) the MILP formulation, (ii) the direct CP model enforcing FIFO on all machines, and (iii) the proposed two-phase CP strategy. Each run is performed with a time limit of 3600 seconds.

[0]The indicator $\mathbb{1}[o_{i,k} = j]$ equals 1 if job i is processed on machine j at position k in its route, and 0 otherwise.

D. Synthetic instance groups for analysis

In addition to benchmarks, we generate synthetic instances to better interpret performance differences between the direct CP and the two-phase CP approaches under controlled structure. Two groups are created with the same $(n, m) = (15, 15)$.

- **Group A (bottleneck-centered).** A machine $j_\star$ is selected and assigned large processing times for all jobs (e.g., uniformly in $[70, 100]$), while processing times on the other machines are sampled from a smaller range (e.g., $[1, 30]$). Moreover, $j_\star$ is forced to appear at a central position in each job route (middle of the routing). This group emphasizes bottleneck-driven behavior and strengthens FIFO-induced propagation through a route-central critical machine.
- **Group B (balanced/random).** Each job route is a random permutation of machines and processing times are sampled uniformly in $[1, 100]$, with no dominant bottleneck. This group represents more homogeneous instances where congestion is not concentrated on a single machine.

We generate 50 instances per group. These two groups are used only to illustrate and explain the observed differences in computational behavior between the CP strategies, complementing the results obtained on standard benchmarks.

V. Experiments and Results

In this section, we report two types of outcomes. First, for instances where optimality is proven within the time limit, we provide the runtime (in seconds) required to certify optimality. Second, for instances where optimality is not proven within $t_{\max} = 3600$s, we compare the best upper bound available at the cutoff time to assess anytime behavior.

a) MILP vs. CP on instances solved by all approaches: On the 37 instances reported in Table I, both CP variants certify optimality much faster than the MILP baseline. Using arithmetic means, the average time-to-proof is 897.38s for MILP, 236.11s for direct CP, and 143.89s for the two-phase CP strategy. Thus, direct CP is on average 3.80 times faster than MILP, and the two-phase CP strategy is 6.24 times faster than MILP on this common solved set.

b) Two-phase CP vs. direct CP on the same set: On the same 37 instances, the two-phase strategy reduces the arithmetic mean runtime from 236.11s (direct CP) to 143.89s, corresponding to a 1.64 times speedup. Instance-wise, two-phase CP is faster on 28 out of 37 instances, while direct CP is faster on the remaining 9 instances.

c) Instances solved by CP within $t_{\max}$ while MILP does not certify optimality: We next consider instances for which at least one CP variant proves optimality within $t_{\max} = 3600$s, whereas the MILP does not reach an optimality proof within the same limit.

For these instances, MILP runtimes are not reported since the MILP solver does not certify optimality within $t_{\max}$.

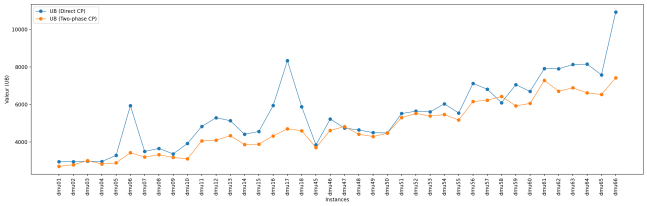
Table II reports the runtimes for the two CP variants.

TABLE I: Classical benchmark instances solved to optimality within $t_{\max} = 3600$s by all approaches. Reported values are runtimes (seconds) to *prove* optimality.

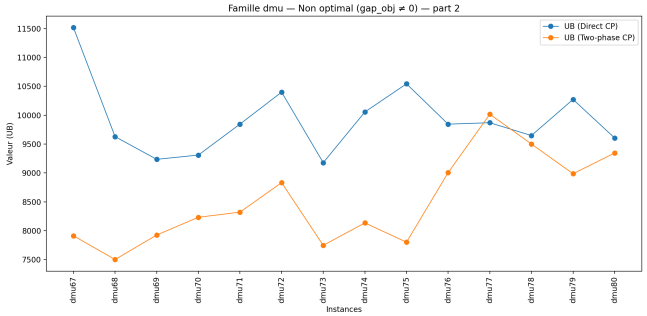
Instances	Objective	MILP (s)	CP direct (s)	two-phase (s)
Adams, Balas and Zawack [1]				
abz5	1242	200.62	4.27	2.65
abz6	945	131.56	2.40	0.52
Fisher and Thompson [5]				
ft06	55	67.12	0.06	0.02
ft10	943	264.32	11.68	32.66
Lawrence [9]				
la01	666	179.50	0.38	0.05
la02	671	179.58	1.37	1.19
la03	618	155.71	0.87	0.36
la04	592	155.34	0.76	1.06
la05	593	145.97	0.15	0.04
la16	954	179.54	4.09	1.57
la17	787	179.59	3.17	1.66
la18	854	179.63	4.25	0.59
la19	865	183.24	10.21	27.55
la20	907	179.61	3.18	1.95
la22	947	266.30	50.38	34.41
la24	962	1540.62	235.88	139.15
la25	998	790.59	165.44	84.47
la36	1278	293.22	78.07	32.54
la39	1250	577.54	81.33	57.04
la40	1237	849.38	329.38	206.60
Applegate and Cook [3]				
orb01	1070	507.89	20.71	14.88
orb02	890	314.03	6.60	8.69
orb03	1021	338.65	19.56	22.81
orb04	1022	301.62	7.63	7.36
orb05	898	315.79	9.30	10.41
orb06	1030	351.88	14.44	34.10
orb07	404	223.92	4.67	14.74
orb08	907	287.29	10.21	1.50
orb09	937	298.93	5.92	3.38
orb10	968	317.29	7.38	7.69
Taillard [14]				
ta02	1265	3530.21	696.41	468.56
ta03	1249	3297.14	505.16	267.23
ta04	1197	2353.82	511.75	206.25
ta05	1247	3521.36	1893.98	1384.97
ta07	1245	3556.04	1425.50	928.31
ta09	1299	3590.31	1469.52	864.69
ta10	1274	3398.03	1140.14	452.45

TABLE II: Instances solved to optimality by CP within $t_{\max} = 3600$s, while the MILP does not prove optimality within the same limit. "–" indicates that direct CP does not prove optimality within $t_{\max}$ (whereas two-phase CP does).

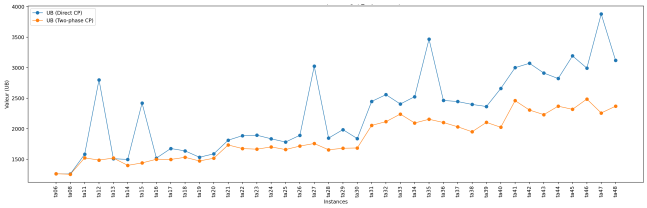
Instances	Objective	CP direct (s)	CP two-phase (s)
dmu32	6076	–	494.05
ft20	1165	9.54	3.35
la06	926	1.25	0.29
la07	890	1.63	5.54
la08	863	2.25	0.20
la09	951	1.13	0.15
la10	958	1.39	0.27
la11	1222	2.63	0.21
la12	1039	3.21	0.46
la13	1150	3.14	0.20
la14	1292	1.66	0.09
la15	1207	124.58	22.91
la21	1070	356.94	274.94
la23	1032	75.59	53.84
la34	1790	–	42.81
la37	1408	88.09	50.25
la38	1222	1888.30	1223.55
swv16	2924	–	72.58
swv17	2794	–	116.84
swv20	2823	–	2218.56



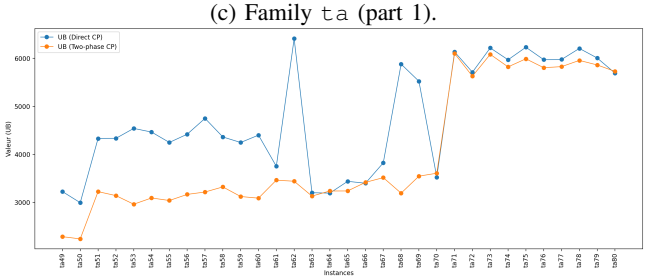
(a) Family `dmu` (part 1).



(b) Family `dmu` (part 2).



(c) Family `ta` (part 1).



(d) Family `ta` (part 2).

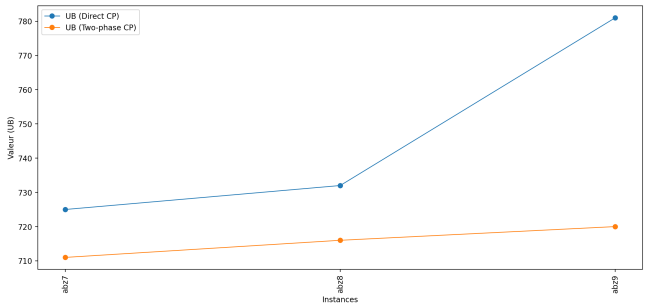
Fig. 1: Upper bounds $UB_i(t_{\max})$ at $t_{\max} = 3600$s.

On the 20 instances reported in Table II, the two-phase CP strategy certifies optimality on all 20 instances, whereas direct CP certifies optimality on only 15/20 instances (it fails within $t_{\max}$ on 5 instances: `dmu32`, `la34`, `swv16`, `swv17`, and `swv20`).

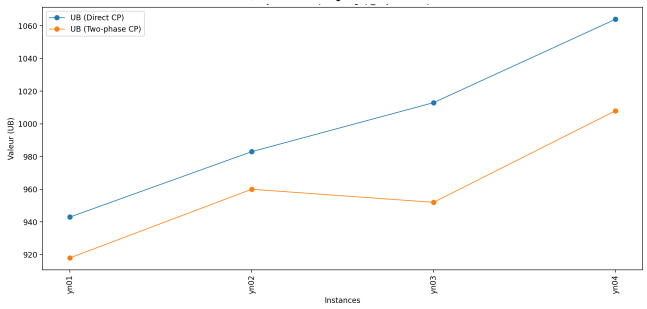
Restricting to the 15 instances solved by both CP variants, the arithmetic mean time-to-proof is 170.76s for direct CP and 109.08s for two-phase CP, corresponding to a 1.57 times speedup.

For instances that do not reach an optimality proof within $t_{\max}$, we compare the best upper bounds (UB) produced at cutoff time. We denote by $UB_i(t_{\max})$ the upper bound obtained for instance i at time $t_{\max}$.

Figures 1–3 provide a complementary view by shifting the focus from proving optimality to solution quality under the time cap. When at least one CP approach does not certify optimality within $t_{\max}$, the relevant indicator becomes

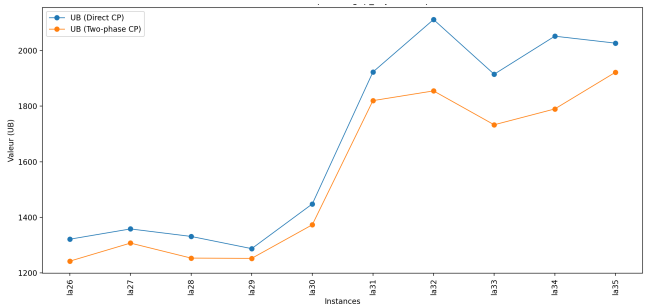


(a) Family `abz`.

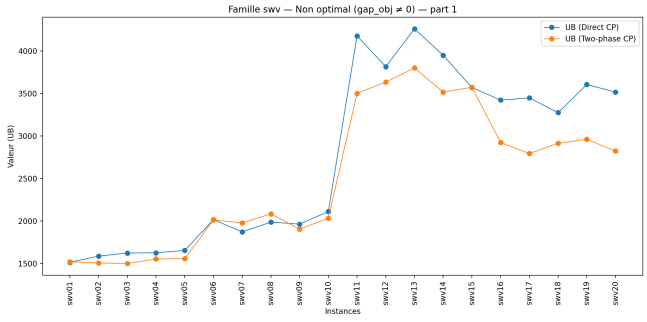


(b) Family `yn`.

Fig. 2: Upper bounds $UB_i(t_{\max})$ at $t_{\max} = 3600$s



(a) Family `la`.

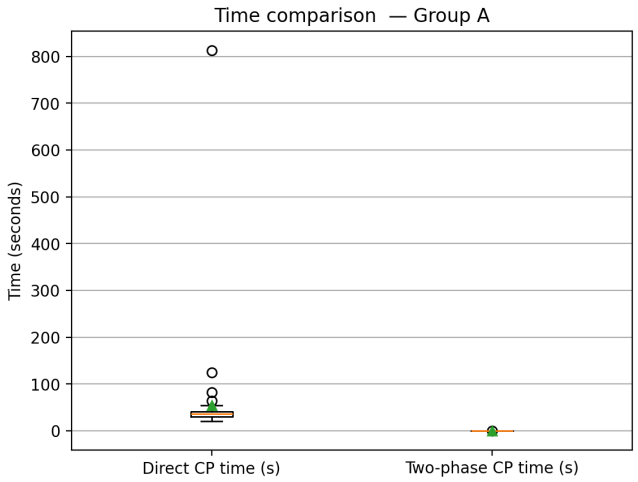


(b) Family `swv`.

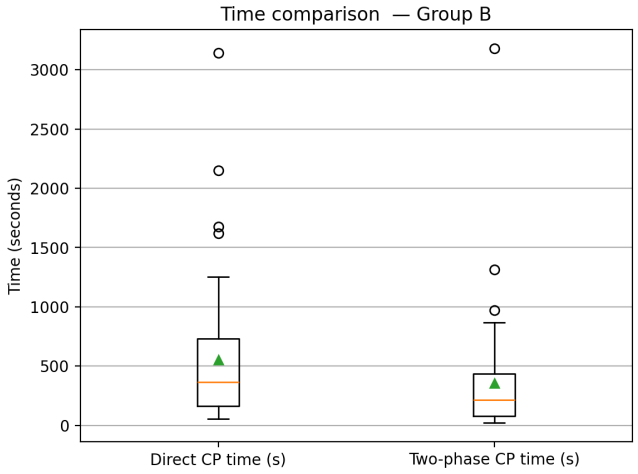
Fig. 3: Upper bounds $UB_i(t_{\max})$ at $t_{\max} = 3600$s.

the best upper bound (UB) obtained at termination. In this regime, the two-phase CP approach attains the tightest UB in the majority of cases.

To interpret the observed runtime differences between the two CP variants, we generated two groups of instances of identical size. Group A is bottleneck-centered, whereas Group B distributes the workload more evenly across machines.



(a) Synthetic Group A (bottleneck-centered).



(b) Synthetic Group B (balanced workload).

Fig. 4: Runtime distributions (boxplots) for direct CP and the two-phase CP approach on synthetic instances.

Figure 4 indicates that the two-phase approach tends to reduce both runtimes and dispersion, with a more pronounced effect on the bottleneck-centered group (Group A). In Group A, the median runtime drops from 36.29s (direct CP) to 0.26s (two-phase), i.e., about 141 times faster, and the interquartile range tightens from 11.26s to 0.10s. In the balanced group (Group B), the improvement is milder but still visible: the median decreases from 363.59s to 214.22s (about 1.70 times faster), and the interquartile range reduces from 571.76s to 356.58s.

VI. Discussion

On the classical benchmark instances from the literature, both CP formulations clearly outperform the MILP baseline in time-to-proof (Table I). This behavior is largely expected: CP is known to be particularly effective on JSP-type problems thanks to strong propagation, global scheduling constraints, and specialized inference mechanisms. Hence, the

observed advantage is consistent with well-established results on classical JSP benchmarks and confirms that CP provides a strong baseline for the FIFO-JSP as well. Comparing the two CP strategies reveals two regimes. On the relatively easier instances in Table I, the two-phase strategy is better on average and faster on most instances, showing that the additional guidance is not merely overhead. Nevertheless, there remain cases where direct CP is faster. A plausible explanation is that, on such instances, the direct model already generates sufficiently strong search guidance early (through propagation and standard branching), so solving an additional Phase 1 problem may not be necessary to obtain a good schedule and strong bounds. The picture changes on harder instances (Table II). In this regime, the two-phase strategy becomes substantially more beneficial: it not only reduces runtimes on the instances solved by both variants, but also certifies optimality on instances where direct CP fails to complete the proof within $t_{\max}$. Moreover, when optimality cannot be certified, solution quality under the time cap becomes critical, and the best upper bound at termination is the appropriate indicator. Figures 1–3 show that the two-phase strategy yields tighter upper bounds in the majority of cases, indicating a more efficient search process under strong FIFO-induced coupling. Overall, these observations suggest that instance structure has a major impact on the relative performance of direct and two-phase CP. The benefit of the two-phase approach is strongest when Phase 1 can exploit a meaningful structural signal (e.g., a bottleneck machine) to impose an informative partial ordering that transfers effectively to the full FIFO model. This intuition motivated our synthetic experiments, which control structural features while keeping the same problem size. As shown in Figure 4, the two-phase method is particularly effective on bottleneck-centered instances, where it reduces both runtime and dispersion, while the gain is more moderate on balanced instances. This supports the interpretation that the two-phase strategy succeeds by leveraging bottleneck structure to produce high-quality guidance and by reducing combinatorial ambiguity when FIFO is restored on all machines. As a limitation, although CP is overall more performant than MILP in our experiments, enforcing FIFO constraints significantly increases difficulty compared to the classical JSP. This suggests that additional guidance mechanisms, alternative formulations, and a sensitivity analysis of the strategic-machine selection rule may be needed to further improve scalability on the most constrained instances.

VII. Conclusion

This study compared exact solution approaches for the FIFO Job Shop Scheduling Problem, including a MILP baseline, a direct CP model, and a two-phase CP strategy. On standard benchmarks, CP achieves markedly lower time-to-proof than the MILP formulation on the instances where all methods certify optimality within the time limit, and it also certifies optimality on additional instances where MILP does not reach an optimality proof under the same budget. Moreover, the two-phase CP strategy improves over direct CP by reducing runtimes on the common solved set and by increasing robustness within the time limit; our analyses also show that the magnitude of this improvement depends strongly on instance structure, with the clearest gains observed on bottleneck-dominated instances. Future work will investigate learning-guided CP search for FIFO-constrained job shop instances, with the aim of improving branching decisions and warm-start quality. Other extensions include soft FIFO variants, where limited FIFO violations are penalized, and multi-objective settings combining makespan with service-quality or waiting-time criteria.

References

[1] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, 1988.

[2] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *European Journal of Operational Research*, vol. 246, no. 2, pp. 345–378, 2015.

[3] D. Applegate and W. Cook, "A computational study of job-shop scheduling," *ORSA Journal of Computing*, vol. 3, no. 2, pp. 149–156, 1991.

[4] J. Carlier and É. Pinson, "An algorithm for solving the job-shop problem," *Management Science*, vol. 35, no. 2, pp. 164–176, 1989.

[5] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*, J. F. Muth and G. L. Thompson, Eds. Englewood Cliffs, NJ, USA: Prentice-Hall, 1963, pp. 225–251.

[6] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[7] V. Heinz, P. Vilím, and Z. Hanzálek, "Reinforcement learning for search tree size minimization in constraint programming: New results on scheduling benchmarks," *Computers & Industrial Engineering*, vol. 209, art. no. 111413, 2025.

[8] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím, "IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG," *Constraints*, vol. 23, no. 2, pp. 210–250, 2018.

[9] S. Lawrence, *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*. Carnegie Mellon University, 1984.

[10] A. S. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. 2, pp. 219–223, 1960.

[11] B. Naderi, R. Ruiz, and V. Roshanaei, "Mixed-integer programming vs. constraint programming for shop scheduling problems: New results and outlook," *INFORMS Journal on Computing*, vol. 35, no. 4, pp. 817–843, 2023.

[12] R. Ouchene, D. Rebaine, and P. Baptiste, "Permutation in shop scheduling problems with FIFO considerations," in *Proc. 11th Int. Conf. on Control, Decision and Information Technologies (CoDIT)*, vol. 1, pp. 1798–1803, IEEE, Jul. 2025.

[13] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, 3rd ed. Springer, 2002.

[14] E. D. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

[15] A. Vela, J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, and I. Amaya, "Beyond hyper-heuristics: A squared hyper-heuristic model for solving job shop scheduling problems," *IEEE Access*, vol. 10, pp. 43981–44007, 2022.

[16] A. Wierman and M. Harchol-Balter, "Classifying scheduling policies with respect to unfairness in an M/GI/1," in *Proc. ACM SIGMETRICS Int. Conf. Measurement and Modeling of Computer Systems*, pp. 238–249, 2003.

[17] W. Y. Ku and J. C. Beck, "Mixed integer programming models for job shop scheduling: A computational analysis," *Computers & Operations Research*, vol. 73, pp. 165–173, 2016.

[18] H. Xiong, S. Shi, D. Ren, and J. Hu, "A survey of job shop scheduling problem: The types and models," *Computers & Operations Research*, vol. 142, art. no. 105731, 2022.