Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 12, n. 1, 2026.

Trabalho apresentado no XLIV CNMAC, Fundação Getulio Vargas - Rio de Janeiro - RJ, 2025.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Mixed-integer versus Constraint Programming Solvers: An Extensive Comparison with the Job Shop Scheduling Problem

Levi R. Abreu[1]
UFC, Fortaleza, CE

Abstract. This paper compares free/open-source and commercial solvers of mixed-integer programming (MILP) and constraint programming (CP) in the classic job shop scheduling problem (JSSP) with makespan and total flowtime minimization. We implemented a MILP model and solved it with CPLEX, Gurobi, and HIGS solvers, and we also implemented and solved CP models with IBM CP, Hexaly, and OR-Tools solvers. We conducted computational experiments using 80 well-known Taillard instance sets. The extensive computational experience shows that the MILP model is promising for solving small-sized instances, and the CP model got the best average relative deviation and superior performance in large-sized instances. The solver IBM CP got the best average results.

Keywords. Production Sequencing, Combinatorial Optimization, OR-Tools, HiGS, Job Shop

1 Introduction

Many scheduling problems are solved using exact and approximate methods, and many of the exact methods are limited to testing mixed-integer linear programming formulations [3]. Constraint programming has emerged as a competitive alternative to model scheduling problems with complex constraints such as no-overlap and precedence [9]. However, there are not many studies comparing MILP and CP approaches, including comparisons between commercial and free/open source solvers [7], which could be an opportunity to leverage the competitiveness of free/open source solvers for use in real-world problems, due to these solvers have a lower practical implementation cost compared to commercial solvers.

The following literature contributions illustrate exact methods applied to the JSSP. Ku and Beck [3] tested new MILP formulations for JSSP with model solutions in just CPLEX, GUROBI, and SCIP solvers. Muller et al. [6] performed comparisons between CP solvers such as IBM CP, OR-Tools, Choco, Chuffed and Geocode for the flexible JSSP problem, with CP Optimizer and OR-Tools obtaining the best performances. The authors do not make any tests with MILP models. Naderi et al. [7] tested only the commercial solvers CPLEX, Gurobi and CP Optimizer on various traditional scheduling problems such as parallel machines, flow shop and job shop. Based on the review, there is a lack of research comparing commercial, free, and open source MILP and CP solvers in a single study, which is applied to scheduling problems in classical instances with the best known solutions.

We study the problem of production scheduling in a job shop environment with minimization of makespan and total flowtime. The main contributions of this paper are the implementation and analysis of MILP and CP models for the JSSP. Subsequently, we conducted extensive experimentation to evaluate the implemented formulations in solving a testbed of 80 Taillard's instances

[1]levi.abreu@ufc.br

DOI: 10.5540/03.2026.012.01.0308

(with the best known solution for makespan minimization in literature and the proposition of best solution for total flowtime in this study) to compare commercial and non-commercial solvers. We evaluate the commercials CPLEX, Gurobi, and the open source/free HiGS solvers for MILP formulation. We also tested the commercials CP Optimizer and Hexaly, and the open source/free OR-Tools solvers for CP formulation.

The remaining sections of this paper are described below. Section 2 addresses the problem statement and some properties. Section 3 addresses the implemented formulations, while Section 4 presents the computational experiments. Finally, in Section 5, we discuss the main findings and suggestions of potential research avenues.

2 Problem Statement

The JSSP consist of the following notations and parameters: let j be an index for jobs {1,2,...,n}, k an index for machine operations for each job {1, 2, ...,m}, and i an index for machines {1,2,...,m}. For each job j there is a matrix p_{ij} with the processing time of job j in machine i and a matrix δ_{kj} with the index of machine of the k^{th} operation of job j. The problem objective is minimizing the last job operation completion time (makespan) or the sum of the last operation completion time of all jobs (total flowtime). Table 1 illustrates an example instance with three machines and five jobs.

Table 1: Data example for the proposed problem.

p_{ij}	J_1	J_2	J_3	J_4	J_5
M_1	5	3	8	2	8
M_2	4	3	6	4	10
M_2	5	3	3	9	9

δ_{kj}	J_1	J_2	J_3	J_4	J_5
k_1	1	1	1	1	2
k_2	2	2	2	3	1
k_3	3	3	3	2	3

A possible solution to the problem presented in Table 1 can be illustrated as a sequence of job operations on each machine. Figure 1 illustrates a Gantt chart with a valid solution of the instance example, with the sequence of jobs on the machines as follows: $M_1 : \{J_4, J_1, J_2, J_5, J_3\}$, $M_2 : \{J_5, J_2, J_1, J_4, J_3\}$ and $M_3 : \{J_4, J_2, J_1, J_5, J_3\}$.

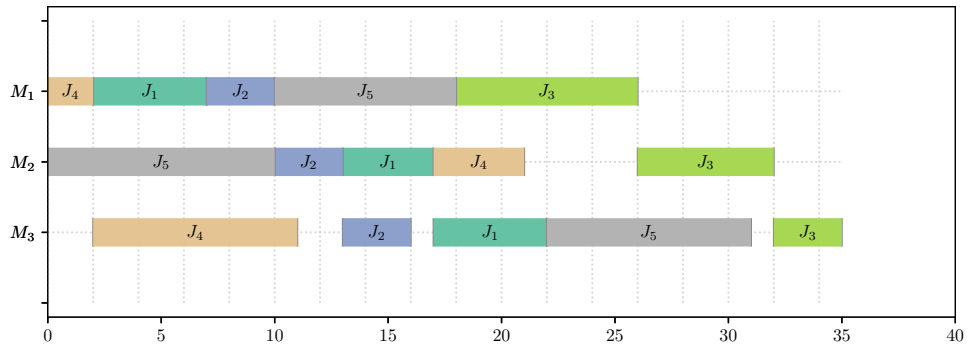


Figure 1: Gantt chart example. Source: authors.

Analyzing Figure 1, we can see the optimum makespan of 35 time units (u.t) and a total flowtime of 126 u.t. To estimate the solution quality of the tested models, we applied the relative percentage deviation (RPD) concerning the current best solution found in the literature for each

instance for makespan minimization and the RPD concerning the best solution found by the solvers for total flowtime minimization, due to there is not available benchmarking in current literature with total flowtime objective function.

3 Developed Formulations

Next, we present the notation, objective function, and constraints used in the MILP and CP models.

3.1 MILP Model

We define the following decision variables for the MILP model: S_{ij} is the start time of the job j in machine i; C_{ij} is the completion time of the job j in machine i; C_{max} is the makespan, and Y_{ijl} is equal to 1 if job j is processed before job l on machine i, 0 otherwise. Finally, $\mathcal{M}$ is a large number used for modeling purposes (the sum of all processing times in matrix p). The MILP model is the formulation proposed by Manne [5].

$$\text{minimize} \quad C_{max} \tag{1}$$

or

$$\text{minimize} \quad \sum_{j-1}^{n} C_{\delta_{mj}j} \tag{2}$$

$$\text{subject to} \quad C_{max} \geq C_{ij}, \qquad \forall i,j \tag{3}$$

$$S_{\delta_{kj}j} \geq S_{\delta_{k-1j}j} + p_{\delta_{k-1j}j}, \qquad \forall k > 1, j \tag{4}$$

$$S_{il} \geq S_{ij} + p_{ij} - \mathcal{M}(1 - Y_{ijl}), \qquad \forall i, l \neq j \tag{5}$$

$$S_{ij} \geq S_{il} + p_{il} - \mathcal{M}Y_{ijl}, \qquad \forall i, l \neq j \tag{6}$$

$$C_{\delta_{kj}j} \geq X_{\delta_{kj}j} + p_{\delta_{kj}j}, \qquad \forall k, j \tag{7}$$

$$C_{max}, C_{ij}, S_{ij} \geq 0, \qquad \forall i,j \tag{8}$$

$$Y_{ijl} \in \{0,1\}, \qquad \forall i,j,l \tag{9}$$

The objective functions (1 and 2) are the makespan minimization or the total tardiness minimization, respectively. Constraints (3) calculate the maximum completion time between all operations. Constraints (4) enforce that the start time of each job operation j must be greater than or equal to the start time of the previous operation plus the processing time. Constraints (5) and (5) determine the non-overlap constraint with the start of an operation in machine i must start after the end of the previous operation. Constraint (7) calculates the completion time in each machine operation i and job j. Finally, constraint sets (8) and (9) determine the domain of the decision variables.

3.2 CP Models

Next, we illustrate the implemented CP model for the problem using the CP Optimizer notation [4]. This CP model uses intervals to represent the operations of jobs in machines and sequence variables to represent sequences of jobs processed in each machine [2, 9]. The CP model uses the same indexes and similar parameters as the MILP model.

As interval variables, $\mathcal{X}_{ij}$ is a unique interval variable for processing the operation of job j in machine i with p_{ij} duration. As a sequence variable, Γ_i is a sequence variable with an order of $\mathcal{X}_{ij}$ interval variables in each machine i. Each interval variable is the sequence of jobs j processed in machine i. The tested CP model is as follows.

DOI: 10.5540/03.2026.012.01.0308

$$\text{minimize} \quad \max_{j\in\{1,2,\ldots,n\}} \texttt{endOf}\left(\mathcal{X}_{\delta_{mjj}}\right) \tag{10}$$

or

$$\text{minimize} \quad \sum_{j=1}^{n} \texttt{endOf}\left(\mathcal{X}_{\delta_{mjj}}\right) \tag{11}$$

$$\text{subject to} \quad \texttt{noOverlap}\left(\Gamma_i\right), \qquad \forall i \tag{12}$$

$$\texttt{endBeforeStart}\left(\mathcal{X}_{\delta_{kjj}}, \mathcal{X}_{\delta_{k+1jj}}\right), \qquad \forall k>1, j \tag{13}$$

$$\text{interval } \mathcal{X}_{ij}, \ \text{ size} = p_{ij}, \qquad \forall i,j \tag{14}$$

$$\text{sequence } \Gamma_i, \ \text{ on } \ \left[\mathcal{X}_{ij}\right]_{i\in\{1,2,\ldots,m\}:j\in\{1,2,\ldots,n\}}, \qquad \forall i \tag{15}$$

The objective functions (10 and 11) are the makespan minimization or the total tardiness minimization, respectively. Constraints (12) impose the non-overlap constraint. Constraints (13) impose the precedence constraint. Finally, constraint sets (14) and (15) define the scope of decision variables. The variable $\mathcal{X}_{ilj}$ is an interval decision variable with a duration, and Γ_i is a variable that stores the sequence of operations on machine i.

The Hexaly and OR-Tools solvers do not use a literature standard modeling language, requiring a complete rewrite of the CP model presented in the IBM CP Optimizer solver. We developed the CP model of the Hexaly solver based on the model presented in `https://www.hexaly.com/docs/last/exampletour/jobshop.html`. We also developed the OR-Tools CP model based on the model presented in `https://developers.google.com/optimization/scheduling/job_shop`.

4 Computational Results

We tested the Taillard instance set for the JSSP [10]. The sets of instances are divided in sizes m x $n \in \{15$ x $15, 15$ x $20, 15$ x $30, 15$ x $50, 20$ x $20, 20$ x $30, 20$ x $50, 20$ x $100\}$. The processing times are uniformly distributed with U[1,99]. Each set of instances has 10 different test problems, totaling 80 base instances.

To compare the methods implemented, we analyze the results of the computational experiments with RPD concerning the best known solution in current literature for makespan minimization and the best solution found by the solvers for total flowtime minimization. Equation (16) shows the RPD calculation of a given instance, where sol_{method} is the value obtained by a method and BKS is the best known solution for the instance. We called the average RPD concerning each set of instances as ARPD (BKS) in all figures and tables in the computational results section.

$$RPD = \frac{sol_{method} - BKS}{BKS} \times 100 \tag{16}$$

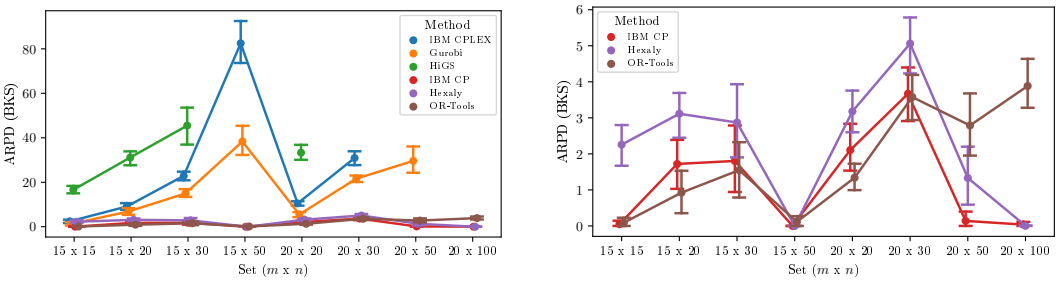
We also reported the percentage of feasible and optimal solution found by each solver (Feasible [%] and Optimal [%] respectively), the average GAP of each solver in relation to the solver's lower bound (Avg. GAP [%]) and the computational time for the solver to run (Avg. Time [s]). The MILP and CP models were run with a 300 time limit in seconds. The MILP models was executed in the IBM CPLEX solver version 22.1.1, GUROBI solver version 11.0.1, and HiGS solver version 1.9.0. The CP model was executed in the IBM CP Optimizer solver version 22.1.1., Hexaly solver version 13.0, and the OR-Tools solver version 9.11. We implemented MILP and CP models in Python 3.11. The computational experience was performed on a PC with Intel Core i7 CPU 4.60 GHz and 16 GB memory, with the Windows 11 operating system. Table 2 illustrates the performance of all solvers with the mentioned indicators for the makespan and total flowtime.

The MILP models were not able to solve the 80 instances for both objectives, and just the IBM CPLEX solver got above 87% of feasible solutions, and HiGS got the best results in Avg.

Table 2: Performance of MILP and CP solvers.

Objective	Solvers	MILP			CP		
		IBM CPLEX	Gurobi	HiGS	IBM CP	Hexaly	OR-Tools
Makespan	Feasible [%]	87.50	81.25	45.00	**100,00**	**100,00**	**100,00**
	Optimal [%]	0.00	2.50	0.00	**46,25**	27,50	**23,75**
	Avg. GAP [%]	44.33	24.81	19.85	3.35	5.01	**3.14**
	ARPD (BKS)	314.69	15.91	30.10	**1.19**	2.23	1.78
	Avg. Time [s]	300.60	298.39	300.02	**197.48**	252.24	258.48
Total Flowtime	Feasible [%]	87,50	87,50	38,75	**100,00**	**100,00**	**100,00**
	Optimal [%]	0,00	0,00	0,00	0,00	0,00	0,00
	Avg. GAP [%]	48,59	26,08	**16,76**	25,03	30,06	22,91
	ARPD (BKS)	121,93	15,21	25,86	**0,49**	1,06	5,61
	Avg. Time [s]	300,47	300,17	300,04	**300,00**	**300,00**	301,66

GAP [%] between MILP models. In addition, all CP models got feasible solutions in each instance tested. Analyzing Table 2, the algorithm with the best ARPD (BKS) is the IBM CP solver in both objectives, which got the best results, outperforming the MILPs and other CP models. The free and open source OR-Tools solver got competitive results similar to IBM CP and got the best Avg. GAP [%] between CP models, denoting the lower bound construction in better than the other models for JSSP. The CP model obtained the best Avg. Time [s], received the best solutions in some instances in a time lower than the time limit of 300 seconds. To check the results of the exact methods in detail, Figure 2a illustrates the ARPD (BKS) in each instance size $m \times n$ for each solver without outliers results (ARPD (BKS) greater than 100%) and Figure 2b lustrates the same analysis only for CP solvers that got better solutions (lower 6% of ARPD (BKS)). Both Figures illustrate results for makespan minimization.



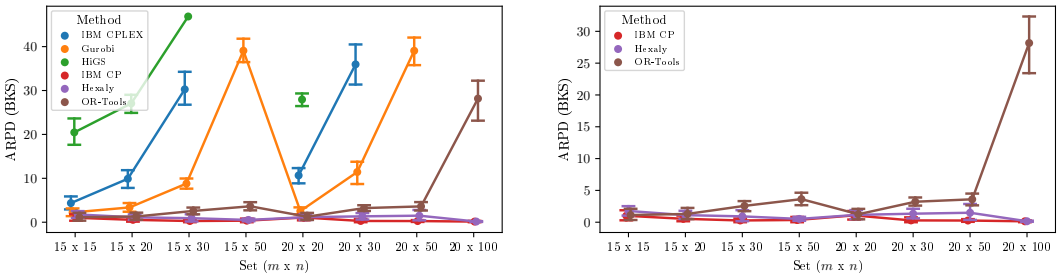
(a) All models.

(b) Only CP models.

Figure 2: ARPD (BKS) in each instance set $m \times n$ for makespan. Source: authors.

Analyzing Figure 2a, the MILP solvers failed to find solutions for instances tested with sizes 20 x 100. Furthermore, for sizes of 15 x 50, the IBM CPLEX solver obtained ARPD (BKS) results greater than about 80%, far from literature best known solution. Analyzing Figure 2b, the IBM CP, Hexaly and OR-Tools model found solutions for all instance sets, producing solutions near from the best known solution from sizes 15 x 50. The OR-Tools solver performed similarly to the commercial IBM CP and Hexaly solvers on instances of sizes less than 20 x 50, even obtaining a better ARPD (BKS) (on average) on instances of size 15 x 20. The IBM CP and Hexaly solvers got ARPD (BKS) close to zero for the largest instances, such as 15 x 50 and 20 x 100. Thus, these solvers provide the best known solutions for larger instances in admissible computational times (up to 300 seconds). Figure 3 shows the same analysis for total flowtime.

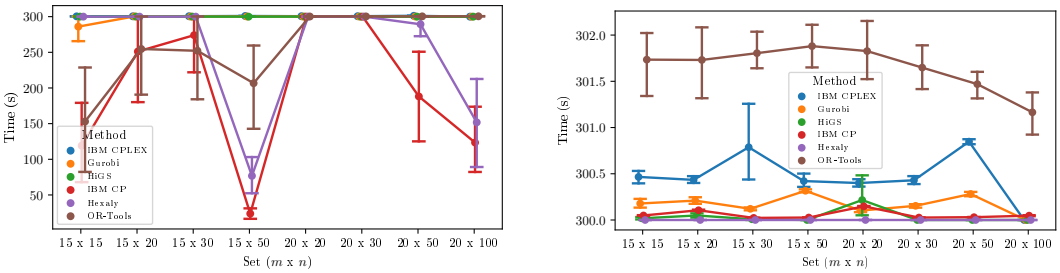
DOI: 10.5540/03.2026.012.01.0308



(a) All models. (b) Only CP models.

Figure 3: ARPD (BKS) results in each instance set $m \times n$ for total flowtime. Source: authors.

Analysis Figure 3, the MILP solvers failed to find solutions for instances tested with the largest sizes. The OR-Tools solver performed similarly to the commercial IBM CP and Hexaly solvers on instances of sizes less than 20 x 50, but performs worse on larger instances. Concerning computational times to solve the instances, Figure 4 shows the average computational time.



(a) Makespan. (b) Total Flowtime.

Figure 4: Avg. Time [s] in each instance set $m \times n$. Source: authors.

In Figure 4, all MILP models reached the time limit for most sets of instances in both objectives. About the CP models in the makespan minimization, the IBM CP and Hexaly solvers obtained the lowest average computational times, especially for large-sized instances, when compared to the other solvers. For the total tardiness minimization, all models reached the time limit.

5 Final Remarks

This paper presented an extensive comparison of exact solvers to the classic JSSP with makespan and total tardiness minimization. We tested the CP and MILP formulations for the problem, considering six different commercial and free/open source solvers. About ARPD (BKS), the MILP solver Gurobi got the best results between MILP solvers. In addition, the CP model executed in the solver IBM CP performed superiorly to the other solvers. However, the free and open source OR-Tools also achieved competitive results similar to IBM CP in the makespan objective. Therefore, the OR-Tools solver becomes a good low-cost alternative in production environments with variants similar to the JSSP, due to its reasonable quality solutions in acceptable computational times.

As extensions of this work, other production scheduling variants could be investigated, such as flow shop or order scheduling, considering different characteristics such as blocking, no-wait and

setup times, or different performance measures, such as the total tardiness. A new execution with a larger time limit could improve the quality of the proposed exact methods. A new standard modeling notation for Hexaly and OR-Tools solvers (as the IBM CP notation) could improve the models formulations in future articles. Finally, developing hybrid algorithms, such as CP matheuristics [1] and metaheuristics with machine learning characteristics [8], are also research opportunities to find feasible solutions in large-sized instances for complex scheduling problems.

Acknowledgment

This work was supported by the Brazilian National Council for Scientific and Technological Development [Grants 404264/2023-9] and the Brazilian Coordination for the Improvement of Higher Education Personnel [Finance Code 001].

References

[1] L. R. de Abreu, K. A. G. Araújo, B. A. de Prata, M. S. Nagano, and J. V. Moccellin. "A new variable neighbourhood search with a constraint programming search strategy for the open shop scheduling problem with operation repetitions". In: **Engineering Optimization** 54.9 (2022), pp. 1563–1582. DOI: 10.1080/0305215X.2021.1957101.

[2] L. R. Abreu, B. A. Prata, M. S. Nagano, and J. M. Framinan. "A constraint programming-based iterated greedy algorithm for the open shop with sequence-dependent processing times and makespan minimization". In: **Computers & Operations Research** 160 (2023). DOI: 10.1016/j.cor.2023.106386.

[3] W. Ku and J. C. Beck. "Mixed integer programming models for job shop scheduling: A computational analysis". In: **Computers & Operations Research** 73 (2016), pp. 165–173. DOI: 10.1016/j.cor.2016.04.006.

[4] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím. "IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG". In: **Constraints** 23 (2018), pp. 210–250.

[5] A. S. Manne. "On the job-shop scheduling problem". In: **Operations research** 8.2 (1960), pp. 219–223.

[6] D. Müller, M. G. Müller, D. Kress, and E. Pesch. "An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning". In: **European Journal of Operational Research** 302.3 (2022), pp. 874–891. DOI: 10.1016/j.ejor.2022.01.034.

[7] B. Naderi, R. Ruiz, and V. Roshanaei. "Mixed-integer programming vs. constraint programming for shop scheduling problems: new results and outlook". In: **INFORMS Journal on Computing** 35.4 (2023), pp. 817–843. DOI: 10.1287/ijoc.2023.1287.

[8] F. B. Ozsoydan. "Reinforcement learning enhanced swarm intelligence and trajectory-based algorithms for parallel machine scheduling problems". In: **Computers & Industrial Engineering** (2025), p. 110948.

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

[10] E. Taillard. "Benchmarks for basic scheduling problems". In: **European Journal of Operational Research** 64.2 (1993), pp. 278–285.

