Multi-objective Optimization of the Job Shop Scheduling Problem on Unrelated Parallel Machines with Sequence-Dependent Setup Times

Francisco dos Santos[1,2(✉)], Lino Costa[1,3], and Leonilde Varela[1,3]

[1] ALGORITMI Research Centre/LASI, University of Minho, Braga, Portugal
francisco_dos_santos@outlook.pt
[2] Polytechnic Institute, University Kimpa Vita, Uíge, Angola
[3] Department of Production and Systems, University of Minho, Braga, Portugal
{lac,leonilde}@dps.uminho.pt

Abstract. Several optimization criteria are involved in the job shop scheduling problems encountered in the engineering area. Multi-objective optimization algorithms are often applied to solve these problems, which become even more complex with the advent of Industry 4.0, mostly due to the increase of data from industrial systems. In this work, several instances of the multi-objective job shop scheduling problem on unrelated parallel machines with sequence-dependent setup times are solved using evolutionary approaches. In this problem, the goal is to assign a set of N jobs on M unrelated machines considering sequence-dependent setup times. Several objectives such as makespan, average completion time, cost and energy consumption can be optimized. In this work, single and multi-objective optimization problems are solved considering the minimization of makespan and the average completion time. Preliminary results for the comparison of algorithms on different instances of the problems are presented and statistically analysed. Future work will include problems with more objectives, and to extend this approach to the distributed job shop problem.

Keywords: Job shop scheduling problem · multi-objective optimization · evolutionary algorithms

1 Introduction

In industry, the area of job shop scheduling (JSS) has been an area of research with a lot of attraction in manufacturing [17]. This is not only a theoretical field of study, but also an interesting field of application in industry and other different real-world situations. Generally, multi-objective optimization problems

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14105, pp. 495–507, 2023.
https://doi.org/10.1007/978-3-031-37108-0_32

are frequent in engineering, and, in particular, in scheduling problems. Thus, it is clear that production problems in practice involve more objectives, and thus it is necessary to extend the solving of single objective JSS problems to multi-objective ones. Makespan, average job completion time, delay, cost savings, latency, energy consumption, and flow time are examples of objectives that comprise a JSS problem in industry [1,16]. Moreover, the task of scheduling different jobs in an optimal sequence is more challenging with the increasing number of jobs, machines, and criteria [13]. This combinatorial optimization problem consists on ordering jobs for their processing on machines taking into account multiple conflicting criteria.

JSS problems can be solved using different algorithms, such as Genetic Algorithms (GAs) [2,9], or, when multiple conflicting objectives exist, NSGA-II, NSGA-III, among other algorithms [10]. These algorithms are particularly suited to this kind of problems since they can tackle their combinatorial nature and multiple conflicting criteria.

This paper addresses the unrelated parallel machine scheduling problem considering the scheduling of N jobs that are available at time zero on M unrelated machines. The objective is to find the feasible sequence of jobs in each machine that minimizes the makespan and the average job completion time considering machine-dependent and sequence dependent setup times [7]. A set of small instances of this problem will be used to assess the GA performance on single objective JSS instances. Based on this analysis, NSGA-II [6] is applied to several multi-objective JSS instances.

This paper is organized as follows. Section 2 briefly presents an overview on multi-objective optimization of JSS, the JSS problem description, and a short presentation about multi-objective optimization concepts and genetic algorithms. In Sect. 3, the experimental results are presented and discussed, highlighting the parameters used in the algorithms, as well as performance measures, and the statistical analysis adopted. Finally, Sect. 4 addresses the conclusion and future work.

2 Multi-objective Job Shop Scheduling

2.1 Overview

JSS problems have been studied for over 50 years, either in academic or industrial settings. Most attention is given to JSS as a single objective optimization problem, which results in a schedule for minimizing the time required to complete all jobs, i.e., minimizing the makespan. Several methods have been developed to solve these problems, such as tabu search [12], ant colony [3], as well as genetic algorithms [14]. However, it is clear that the complexity of JSS problems besides being related to the dimension in terms of number of jobs and machines, depends also on the number of objectives and constraints.

JSS problems, in general, involve many objectives. NSGA-II has been applied to solve a set of instances of the flexible JSS problem [15]. In order to improve the efficiency, these authors implemented a grid representation of the solution space

and concurrent priority queue to store and dispatch the pending sub-problems to be solved.

There are different representations of the solution space that can be used by GAs for solving JSS problems. These different representation affect application of genetic operators such as recombination and mutation and therefore the performance of the algorithms. In [1], it is provided a computational study to compare the performance of a GA under different representations.

An extension of the NSGA-III algorithm (i-NSGA-III) is proposed in [10] to solve manufacturing JSS problems with multiple objectives. These authors also developed novel problem-specific crossover and mutation operators. The results demonstrated the superiority of the extended version over the original NSGA-III for several instances of multi-objective JSS problems.

2.2 Problem Description

The unrelated parallel machine scheduling problem consists on the scheduling of N jobs on M unrelated machines. The term span is generally used to define the completion time of a given machine, while the term makespan is generally used for the maximum completion time [2]. In spite of makespan being the most common objective to be minimized, there are other relevant conflicting objectives such as the average completion time [16].

The following nomenclature is used for the JSS problem:

- π is the feasible sequence of all jobs assigned to machines, i.e., the vector of all feasible subsequences π_i on each machine i;
- π_i is the feasible subsequence of jobs on machine i;
- $p_{i,j}$ is the processing time of job j on machine i;
- $s_{i,j,0}$ is the setup time to process job j at the initial instant on machine i;
- $s_{i,j,k}$ is the setup time to process job j after job k on machine i;
- $c_i(\pi)$ is the total completion time of jobs on machine i for the feasible sequence π.

The following assumptions are considered in this paper:

- each machine M can only execute one job N at each instant;
- all machines are continuously available;
- all N jobs are independent without any precedence restriction;
- all N jobs are available for being executed at the initial instant;
- no preemption is allowed, i.e., the processing of a job in a machine cannot be interrupted;
- for each machine i, each job j has a processing time $p_{i,j}$;
- for each machine i, to process job j after job k, there is a setup time $s_{i,j,k}$.

In this paper, the goal is to find the feasible sequence of jobs in each machine that minimizes simultaneously the makespan [1] and the average job completion time [8] considering machine-dependent and sequence dependent setup times [7].

Thus, the objectives to minimize are the makespan ($c_{max}(\pi)$) and the average completion time ($\bar{c}(\pi)$).

Mathematically, the makespan and the average completion time can be formulated as:

$$c_{\max}(\pi) = \max_{i=1,\ldots,M}(c_i(\pi)) \tag{1}$$

and

$$\bar{c}(\pi) = \frac{1}{M}\sum_{i=1}^{M} c_i(\pi) \tag{2}$$

where $c_i(\pi) = \sum_{j\in\pi_i} p_{i,j} + \sum_{j\in\pi_i} s_{i,j,k}$ with k being the precedent job of job j in the subsequence π_i of the feasible sequence π. Note that if the job j is the first job in the subsequence π_i then k is zero and the setup time $s_{i,j,0}$ at the initial instant is considered.

2.3 Multi-objective Optimization

A multi-objective optimization problem with K objectives and n decision variables can be formulated mathematically as follows:

$$\min \mathbf{F}(\mathbf{x}) = (f_1(x), f_2(x), \ldots, f_K(x))^T \tag{3}$$

where $\mathbf{x}$ is the decision vector of size n defined in the decision space Ω, and $\mathbf{F}(\mathbf{x})$ is the objective vector defined in the objective space.

When several conflicting objectives are optimized at the same time, the search space becomes partially ordered. In such scenario, solutions are compared on the basis of the Pareto dominance. Without loss of generality, consider a multi-objective optimization problem where K objectives are to be minimized, for two solutions $\mathbf{a}$ and $\mathbf{b}$ from the feasible set Ω, a solution $\mathbf{a}$ is said to dominate a solution $\mathbf{b}$ (denoted by $\mathbf{a} \prec \mathbf{b}$) if:

$$\forall_{i\in\{1,\ldots,K\}} : f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \wedge \exists_{j\in\{1,\ldots,K\}} : f_j(\mathbf{a}) < f_j(\mathbf{b}). \tag{4}$$

Since solutions are compared against different objectives, there is no longer a single optimal solution but a set of optimal solutions, generally known as the Pareto optimal set. This set contains equally important solutions representing different trade-offs between the given objectives and can be defined as:

$$\mathbf{PS} = \{\mathbf{x} \in \Omega \mid \nexists \mathbf{y} \in \Omega : \mathbf{y} \prec \mathbf{x}\}. \tag{5}$$

The images of the solutions of the Pareto optimal set define a Pareto front in the objective space. This Pareto front allow to identify the trade-offs between solutions and therefore facilitate the decision making.

2.4 Genetic Algorithms

A Genetic Algorithm (GA) is a random search optimization technique that mimics the process of natural selection [5]. It is based on a population that it is generated at random in the beginning of the search. The individuals of the population are, in general, referred as chromosomes and represent potential solutions of the optimization problem. A selection mechanism is used to choose solutions to generate offspring. This mechanism assures that the best chromosomes have higher probabilities of being selected. Genetic operators such as crossover and mutation allow to create new chromosomes that inherit features from the parents. Recombination generates two new child solutions by mating the selected chromosomes, exchanging of specific genes of the parents. Mutation introduces some new elements in the chromosomes in order to increase their ability to reach other points in the search space.

One of the relevant advantages of GAs is that they can work with a population of chromosomes that represent potential solutions of the optimization problem and are adequate to tackle combinatorial problems, since the constraints can be handled by permutation representations. On the other hand, GAs are particularly suited to solve multi-objective optimization problems since they work with population of solutions and therefore can find in a single run an approximation to the Pareto optimal set. Furthermore, diversity-preserving mechanisms can be implemented to spread solutions along the Pareto front to provide a set of good trade-offs to facilitate decision making.

3 Experimental Results

3.1 Test Problems

The test problems were borrowed from previous works [2,3]. These problems were used to investigate the GA performance and the quality of the NSGA-II results. In this study, from the set of all available problems, only the small problems with balanced processing and setup times for $M \in \{2, 4, 6, 8\}$, and $N \in \{6, 7, 8\}$ were considered. For each combination of the number of machines and jobs, there are 15 different instances of the problems. First, a comparative and statistical analysis of the solutions obtained by GA for the single objective optimization problems considering the makespan and the average completion time, separately, is presented. Then, NSGA-II was applied to solve multi-objective optimization problems considering the makespan and the average completion time, simultaneously.

3.2 Algorithms Implementation and Parameters

In this work, two approaches based on genetic algorithms, GA and NSGA-II, were used to solve the single and multiple-objective JSS optimization problem, respectively. These algorithms were implemented in Python using the PYMOO framework [4].

A permutation representation was adopted to represent the feasible sequences of jobs in each machine (π). A solution is represented by a chromosome that is a permutation of size $N + M - 1$, i.e., a sequence of integer values that can occur only once. In a chromosome, the genes that are superior to N separate the chromosome in M subsequences (π_i) that indicate the set of jobs and the corresponding order assigned to each machine i.

A tournament selection operator was used to select chromosomes for the application of genetic operators. Genetic operators were chosen to guarantee feasibility of the solutions during the search: the order-based crossover and inversion mutation. In the order-based crossover, two chromosomes are combined in order to generate offspring. Two positions are selected at random and the genes between them are swapped. Then, the remaining empty positions are filled with the other parent's genes while preventing repetitions. In the inversion mutation, two positions are generated at random, and the genes in the chromosome are inverted.

In order to find an adequate population size, two different population sizes with 20 and 40 individuals were tested to solve the single objective optimization problems with GA. For the multi-objective optimization problems, NSGA-II was applied with the population size that achieved the best GA performance. The stopping criterion was based on the maximum number of generations and the period of no improvement. In this case, the total number of generations was set to 10,000 and the period to 100 generations. Since the algorithms used are stochastic, for all instances of all test problems, 30 independent runs were performed. In this manner, the results can be statistically analysed. For all statistics tests, a significance level of 5% is considered.

3.3 Single Objective Optimization Results

In this section, the results of the application of GA to single objective JSS optimization problems are presented. Two objectives are considered separately: the makespan and the average completion time. The performance of GA with a population size of 20 and 40 individuals is statistically analysed.

In Table 1, the statistics for the GA performance on single objective problems for the minimization of the makespan for a population size of 20 and 40 individuals is presented. In this table, $AVG(c_{\max}(\pi))_{40}$ $SD(c_{\max}(\pi))_{40}$ and $AVG(c_{\max}(\pi))_{20}$ $SD(c_{\max}(\pi))_{20}$ are the average and standard deviation of the makespan ($c_{\max}(\pi)$) for a population size of 40 and 20 individuals, respectively. These statistics are computed for every combination of M (the number of machines) and N (the number of jobs). For each combination of M and N, the average values of 30 independent runs are calculated and then the statistics (average and standard deviation) for the 15 instances are computed. The p-values for the one-tailed paired t-Student test are also provided [11].

It can be observed that, for all combinations of M and N, the p-values are inferior to 5%, then there are significant differences between the mean values of $c_{\max}(\pi)$ for population sizes of 40 and 20 individuals. Thus, the GA performance with a population size of 40 individuals was significantly superior to the

Table 1. Single objective results for the minimization of the makespan ($c_{\max}(\pi)$) for the GA with a population size of 40 and 20 individuals.

M	N	$AVG(c_{\max}(\pi))_{40}$	$SD(c_{\max}(\pi))_{40}$	$AVG(c_{\max}(\pi))_{20}$	$SD(c_{\max}(\pi))_{20}$	p-value
2	6	394.773	14.168	395.049	14.173	0.020
	7	491.807	15.805	493.240	15.643	<0.001
	8	521.961	17.087	524.823	17.454	<0.001
4	6	246.220	7.092	247.782	6.850	0.016
	7	255.683	9.676	257.577	8.933	<0.001
	8	270.063	6.301	272.371	5.968	<0.001
6	8	239.093	5.472	241.280	6.157	<0.001

algorithm with a population size of 20 individuals for the minimization of the makespan ($c_{\max}(\pi)$).

For illustration purposes, Fig. 1 represents the optimal solution obtained for instance 5 of the minimization of the makespan with 2 machines (m_1 and m_2) and 6 jobs (n_1, n_2, n_3, n_4, n_5 and n_6). In this figure, blue boxes represent the jobs. Additionally, the processing times and setup times are also indicated below the blue boxes and before the jobs, respectively. For instance 5, the processing times $p_{i,j}$ of the job j on machine i are given in the following matrix:

$$p = \begin{pmatrix} 83 & 77 & 55 & 99 & 86 & 74 \\ 53 & 62 & 90 & 60 & 52 & 83 \end{pmatrix}.$$

The setup times $s_{i,j,k}$ to process job j after job k are given on matrices s_1 and s_2 for machines m_1 and m_2, respectively:

$$s_1 = \begin{pmatrix} 77 & 91 & 82 & 95 & 50 & 93 \\ 52 & 70 & 94 & 71 & 58 & 85 \\ 67 & 83 & 77 & 69 & 70 & 69 \\ 85 & 68 & 51 & 80 & 58 & 99 \\ 62 & 66 & 53 & 53 & 64 & 98 \\ 74 & 76 & 82 & 58 & 86 & 97 \end{pmatrix} \quad s_2 = \begin{pmatrix} 60 & 72 & 88 & 73 & 60 & 85 \\ 61 & 82 & 53 & 75 & 65 & 53 \\ 68 & 51 & 52 & 58 & 86 & 96 \\ 98 & 69 & 51 & 99 & 71 & 58 \\ 76 & 55 & 73 & 77 & 94 & 85 \\ 98 & 84 & 92 & 79 & 51 & 89 \end{pmatrix}.$$

In the diagonal of the matrices, the setup time to process job j at the initial instant on machines ($s_{i,j,0}$) are given. This solution corresponds to the sequence $\pi = <\pi_1, \pi_2>$ where $\pi_1 = [2, 5, 3]$ and $\pi_2 = [1, 4, 6]$. The total completion times are $c_1(\pi) = 399$ and $c_2(\pi) = 387$, and therefore $c_{\max}(\pi) = 399$.

In Table 2, the statistics for the GA performance on single objective problems for the minimization of the average completion time for a population size of 20 and 40 individuals is presented. As previously, in this table, the averages and standard deviations of the average completion time ($\bar{c}(\pi)$) for a population size of 20 and 40 individuals are indicated. The statistics and p-values presented in this table were computed similarly as those presented in Table 1.

It can be observed that, for all combinations of M and N, the p-values are inferior to 5%. Thus, the GA performance with a population size of 40

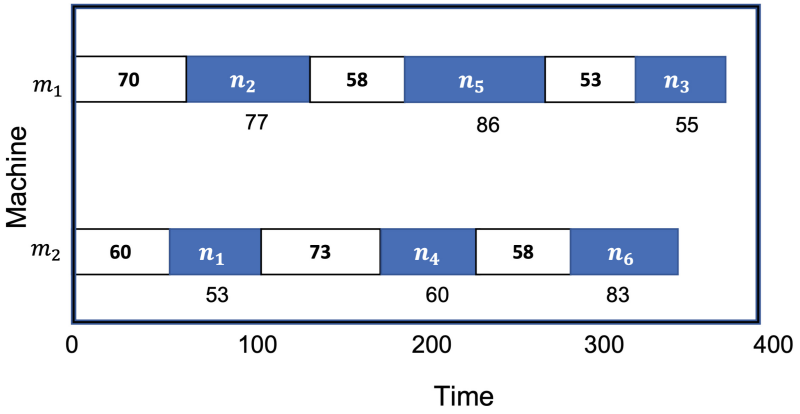

Fig. 1. Representation of the optimal solution ($c_{\max} = 399$) obtained for the minimization of the makespan with 2 machines and 6 jobs (instance 5).

Table 2. Single objective results for the minimization of the average completion time ($\bar{c}(\pi)$) for the GA with a population size of 40 and 20 individuals.

M	N	$AVG(\bar{c}(\pi))_{40}$	$SD(\bar{c}(\pi))_{40}$	$AVG(\bar{c}(\pi))_{20}$	$SD(\bar{c}(\pi))_{20}$	p-value
2	6	378.816	12.920	378.979	12.910	0.042
	7	446.227	12.700	447.012	12.627	0.002
	8	504.145	15.717	504.883	15.792	<0.001
4	6	182.536	5.104	183.017	3.873	<0.001
	7	211.641	6.255	212.637	5.928	0.003
	8	240.680	4.525	242.259	4.517	<0.001
6	8	159.232	4.153	160.397	3.951	<0.001

individuals was significantly superior to the algorithm with a population size of 20 individuals for the minimization of the average completion time ($\bar{c}(\pi)$).

Figure 2 represents the optimal solution obtained for instance 5 of the minimization of the average completion time with 2 machines and 6 jobs. This solution corresponds to the sequence $\pi = <\pi_1, \pi_2>$ where $\pi_1 = []$ and $\pi_2 = [3, 4, 6, 5, 2, 1]$. The total completion times are $c_1(\pi) = 0$ and $c_2(\pi) = 735$, and therefore $\bar{c}(\pi) = 367.5$.

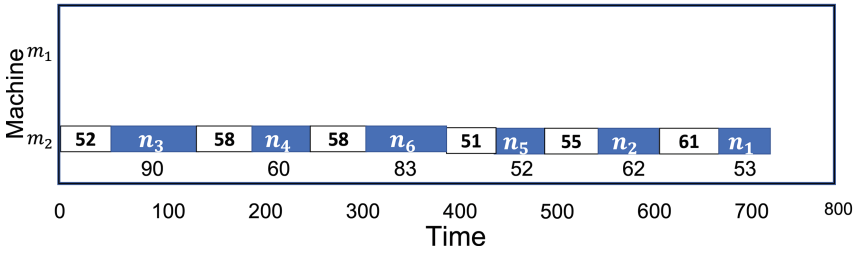

Fig. 2. Representation of the optimal solution ($\bar{c}(\pi) = 367.5$) obtained for the minimization of the average completion time with 2 machines and 6 jobs (instance 5).

For measuring the relative improvement in the GA performance in terms of the objective function values, a δ measure was computed for different statistics. Thus, a δ_{sta} measure was computed according to the following expression:

$$\delta_{sta} = \frac{obj_{40} - obj_{20}}{obj_{40}}$$

where *sta* and *obj* are the statistic and the objective considered, respectively. If the δ measure is negative then the best results were obtained by the GA for a population size of 40 individuals. The opposite occurs when the δ measure is positive. If the δ measure is equal to zero then the performance is similar.

In Table 3, the statistics of the δ measure for the comparison of GA with 40 and 20 individuals for the makespan and average completion time objectives is given. The δ measure was computed for the best objective function results ($\delta_{\min}$) and the average objective function results (δ_{mean}).

Taking into account the results presented in Table 3, for both $c_{\max}(\pi)$ and $\bar{c}(\pi)$, δ_{mean} is always negative and the p-values are all inferior to 5%, meaning that the GA with a population size of 40 individuals significantly achieved a better performance when compared to the GA with a population size of 20. However, for $\delta_{\min}$, no significant differences are found for $c_{\max}(\pi)$. Finally, for $\bar{c}(\pi)$, no significant differences are found for all combinations of M and N except for $M = 2$ and $N = 7$, and $M = 2$ and $N = 8$. Overall, these results indicate a better performance for a population size of 40 individuals. Moreover, this results in a significant relative improvement of average objective values.

Table 3. Statistics of the δ measure for the comparison of GA with 40 and 20 individuals for the makespan and average completion time objectives.

M	N	$c_{\max}(\pi)$				$\bar{c}(\pi)$			
		$AVG(\delta_{\min})$	p-value	$AVG(\delta_{\text{mean}})$	p-value	$AVG(\delta_{\min})$	p-value	$AVG(\delta_{\text{mean}})$	p-value
2	6	0.0000	1.000	−0.0007	0.020	0.0000	1.000	−0.0004	0.040
	7	0.0000	1.000	−0.0029	<0.001	0.0004	0.004	−0.0018	0.002
	8	0.0000	1.000	−0.0055	<0.001	0.0005	0.002	−0.0015	<0.001
4	6	0.0000	1.000	−0.0064	0.016	0.0005	1.000	−0.0026	<0.001
	7	0.0000	1.000	−0.0075	<0.001	−0.0003	0.321	−0.0048	0.003
	8	0.0048	0.167	−0.0086	<0.001	0.0000	1.000	−0.0066	<0.001
6	8	−0.0014	0.167	−0.0091	<0.001	−0.0002	0.305	−0.0074	<0.001

3.4 Multi-objective Optimization Results

In this section, the results of the application of NSGA-II to multi-objective JSS optimization problems are presented. Two objectives are considered simultaneously: the makespan and the average completion time.

The hypervolume indicator (H) was used to measure the performance of the NSGA-II [18]. This measure evaluates algorithm performance in terms of convergence to the Pareto front as well as the diversity of the approximation along the Pareto front. The reference point considered was the point $(1000, 1000)^T$ and all hypervolume values were standardized. Thus, all hypervolume values have values belonging to the interval $[0, 1]$. The higher the value of hypervolume, the best performance of the algorithm.

In Fig. 3, the Pareto front obtained for the minimization of the makespan and the average completion time with 2 machines and 6 jobs (instance 5) is depicted. In this figure, it is possible to identify the trade-offs between the objectives for different solutions. The extreme solutions in the Pareto front are the solutions obtained when the single objective JSS problems were solved. Thus, the solutions on the upper-left and lower-right extremes of the Pareto front correspond to solutions presented in Fig. 1 and Fig. 2.

Figure 4 represents a trade-off solution that is a good compromise between the objectives. This solution is in the elbow of the Pareto curve. This solution corresponds to the sequence $\pi = <\pi_1, \pi_2>$ where $\pi_1 = [3, 6]$ and $\pi_2 = [1, 5, 2, 4]$. The total completion times are $c_1(\pi) = 275$ and $c_2(\pi) = 477$, and therefore $c_{\max}(\pi) = 477$ and $\bar{c}(\pi) = 376$. that are in the elbow of the curve. This solution is a good compromise between the objectives.

In Table 4, the statistics for the measured hypervolume for Pareto fronts obtained by NSGA-II for 15 instances of the problems for each combination of M and N are presented. In this table, $AVG(H_{\max})$, $SD(H_{\max})$ are the average and standard deviation of the best hypervolume (maximum hypervolume), and $AVG(H_{\text{mean}})$, $SD(H_{\text{mean}})$ are the average and standard deviation of the mean hypervolume, respectively. In order to compare this multi-objective results with the single objective ones, the values of $AVG(H_{ref})$ and $SD(H_{ref})$ were also included in this table. These values are the average and standard deviation of the hypervolume computed for the minimum of each objective (the extremes of the Pareto font). It can be seen that the value of $AVG(H_{\max})$ is always superior to the value $AVG(H_{ref})$. This highlights the importance of using multi-objective approaches. The values of $AVG(H)$ are for some combinations of M and N inferior to $AVG(H_{ref})$. However, it should be noted that this occurs due to the fact that, for some instances, in a small number of the 30 runs, the approximation obtained is not so close to the optimal Pareto front. Thus, the hypervolume average values are affected by these cases.

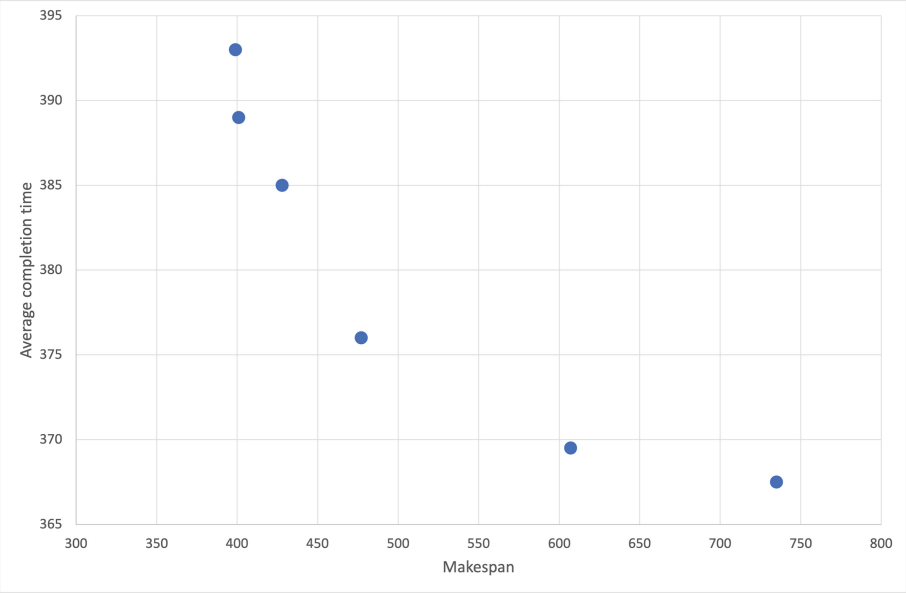

Fig. 3. Pareto front for the minimization of the makespan and the average completion time with 2 machines and 6 jobs (instance 5).

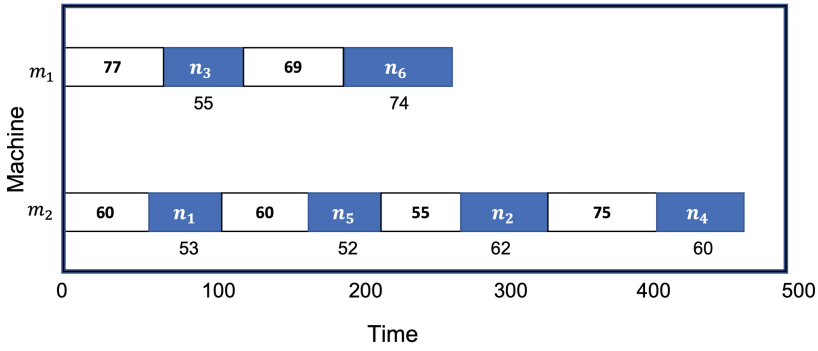

Fig. 4. Representation of a trade-off solution ($c_{\max}(\pi) = 477$ and $\bar{c}(\pi) = 376$) obtained for the minimization of makespan and the average completion time with 2 machines and 6 jobs (instance 5).

Table 4. Statistics for the measured hypervolume for Pareto fronts on multi-objective JSS problems.

M	N	$AVG(H_{\max})$	$SD(H_{\max})$	$AVG(H)$	$SD(H)$	$AVG(H_{ref})$	$SD(H_{ref})$
2	6	0.375	0.016	0.375	0.016	0.374	0.016
	7	0.281	0.014	0.280	0.014	0.280	0.014
	8	0.239	0.016	0.237	0.017	0.238	0.016
4	6	0.617	0.008	0.615	0.008	0.616	0.008
	7	0.589	0.012	0.586	0.011	0.586	0.012
	8	0.558	0.008	0.553	0.009	0.556	0.008
6	8	0.644	0.008	0.640	0.008	0.643	0.009

4 Conclusions and Future Work

In this paper, several instances of JSS problems on unrelated parallel machines with sequence-dependent setup times were solved as single objective and multi-objective optimization problems. These problems were solved using evolutionary algorithms for different number of machines and jobs. The makespan ($c_{\max}$) and the average completion time ($\bar{c}$) were the objectives considered.

Firstly, a GA algorithm was used to solve instances of the single objective optimization problems consisting on the minimization of the objectives separately. The results of several preliminary tests were analysed to tune the GA parameters, namely, the population size. Then, taking into consideration these results, instances of the multi-objective optimization problem were solved using NSGA-II. In all cases, a permutation was used to represent the sequence of jobs in each machine. All results were statistically analysed. It was concluded that using small population sizes can compromise the performance of the algorithms in this kind of problems. The best results were obtained with a population size of 40 individuals.

The analysis of the hypervolume results for the multi-objective JSS problems confirmed the advantages and importance of using multi-objective approaches since that indicate the existence of conflict between the objectives and multiple trade-offs. The representation of the Pareto front highlights the trade-offs facilitating the decision making process.

Future work will include solving problems with more objectives and the use of other algorithms, and to extend this approach to the distributed job shop problem.

References

1. Abdelmaguid, T.F.: Representations in genetic algorithm for the job shop scheduling problem: a computational study. J. Softw. Eng. Appl. **3**(12), 1155 (2010)
2. Antunes, A.R., Matos, M.A., Rocha, A.M.A., Costa, L.A., Varela, L.R.: A statistical comparison of metaheuristics for unrelated parallel machine scheduling problems with setup times. Mathematics **10**(14), 2431 (2022)

3. Arnaout, J.P., Musa, R., Rabadi, G.: A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines-part II: enhancements and experimentations. J. Intell. Manuf. **25**, 43–53 (2014)
4. Blank, J., Deb, K.: Pymoo: multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
5. Deb, K.: An introduction to genetic algorithms. Sadhana **24**, 293–315 (1999)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
7. Feldman, A.: Scheduling algorithms and systems. Columbia University (1999)
8. González-Neira, E.M., et al.: Robust solutions in multi-objective stochastic permutation flow shop problem. Comput. Industr. Eng. **137**, 106026 (2019)
9. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimed. Tools Appl. **80**, 8091–8126 (2021)
10. Khan, B., Hanoun, S., Johnstone, M., Lim, C.P., Creighton, D., Nahavandi, S.: Multi-objective job shop scheduling using I-NSGA-III. In: 2018 Annual IEEE International Systems Conference (SysCon), pp. 1–5. IEEE (2018)
11. Montgomery, D.C., Runger, G.C.: Applied Statistics and Probability for Engineers. Wiley, Hoboken (2010)
12. Saidi-Mehrabad, M., Fattahi, P.: Flexible job shop scheduling with tabu search algorithms. Int. J. Adv. Manuf. Technol. **32**, 563–570 (2007)
13. dos Santos, F., Costa, L.A., Varela, L.: A systematic literature review about multi-objective optimization for distributed manufacturing scheduling in the industry 4.0. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds.) ICCSA 2022, Part II. LNCS, vol. 13378, pp. 157–173. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10562-3_12
14. Sha, D., Lin, H.H.: A multi-objective PSO for job-shop scheduling problems. Expert Syst. Appl. **37**(2), 1065–1070 (2010)
15. Soto, C., Dorronsoro, B., Fraire, H., Cruz-Reyes, L., Gomez-Santillan, C., Rangel, N.: Solving the multi-objective flexible job shop scheduling problem with a novel parallel branch and bound algorithm. Swarm Evol. Comput. **53**, 100632 (2020)
16. Yenisey, M.M., Yagmahan, B.: Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega **45**, 119–135 (2014)
17. Zhang, J., Ding, G., Zou, Y., Qin, S., Fu, J.: Review of job shop scheduling research and its new perspectives under industry 4.0. J. Intell. Manuf. **30**, 1809–1830 (2019)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)