

# Multi-objective Scheduling Optimization in Job Shop with Unrelated Parallel Machines Using NSGA-III

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, Kimpa Vita University, Uíge, Angola
[3] Department of Production and Systems, University of Minho, Braga, Portugal
{lac,leonilde}@dps.uminho.pt

**Abstract.** Job shop scheduling problems are common in the engineering field. In spite of some approaches consider just the most important objective to optimize, several other conflicting criteria are also important. Multi-objective optimization algorithms can be used to solve these problems optimizing, simultaneously, two or more objectives. However, when the number of objectives increases, the problems become more challenging. This paper presents the results of the optimization of a set of job shop scheduling with unrelated parallel machines and sequence-dependent setup times, using the NSGA-III. Several instances with different sizes in terms of number of jobs and machines are considered. The goal is to assign jobs to machines in order to simultaneously minimize the maximum job completion time (makespan), the average job completion time and the standard deviation of the job completion time. These results are analysed and confirm the validity and highlight the advantages of this approach.

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

## 1 Introduction

In general, Job Shop Scheduling (JSS) has played a fundamental role in manufacturing companies [20], as it maximizes efficiency while minimizing various costs, delivery time and cycle time, delays, among others [13]. Several studies have been carried out in order to analyse the impact of the job shop scheduling problem on manufacturing systems [1,14,19]. JSS problems with unrelated parallel machines aim to schedule $N$ jobs that are available at initial time on $M$ unrelated machines [9]. The goal is to find the feasible sequence of jobs in

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 2024
O. Gervasi et al. (Eds.): ICCSA 2024 Workshops, LNCS 14816, pp. 370–382, 2024.
https://doi.org/10.1007/978-3-031-65223-3_25

each machine that minimizes simultaneously the objectives. Thus, JSS considers different machines with different sequences and processing times.

Most studies addressing JSS problems consider just the total production time (makespan) as the objective to minimize [2,12]. Therefore, these problems are solved using single objective optimization algorithms, often, based on metaheuristics [3,4]. Although, makespan being one of the main objectives to be minimized in the context of production, there are other objectives that can be simultaneously minimized, thus improving the optimization results [10,16]. It is important to maximize profits and balance workloads by minimizing different objectives such as the makespan, the average and standard deviation production times between machines. Therefore, it is required to use multi-objective optimization algorithms to solve this problem with more than two objectives. Multi-objective scheduling problems are common in industry and are considered complex because they involve different objectives to be optimized simultaneously as well as a combinatorial search space [17].

This paper presents a multi-objective optimization approach for the JSS problem, aiming to simultaneously minimize tree objectives: the total completion time of the jobs, the average completion time of the jobs and the standard deviation of the total completion time of the jobs. To solve this kind of problems, there are several multi-objective optimization algorithms studied and available in the literature [5], most of them based on evolutionary algorithms. In this work, the NSGA-III [7] was adopted, as it performs better when the number of objectives to be minimized is more than two [18]. NSGA-III is particularly suited to solve problems with many objectives allowing to obtain a good representation of the Pareto front since the search is oriented by a certain number of well distributed reference points [11,13]. Moreover, evolutionary algorithms are particularly suited to this kind of problems since they can tackle their combinatorial nature. In order to assess the performance of NSGA-III and compare it with other approaches, a set of small JSS instances are considered [3,4].

This paper is organized as follows. Section 2 briefly presents an overview on multi-objective optimization of JSS, describing the optimization problem and the multi-objective approach adopted in this work. In Sect. 3, the experimental results of NSGA-III are presented and discussed. In this section, the implementation details and parameters of NSGA-III are explained, as well as the analysis of its overall performance. Finally, Sect. 4 addresses the conclusion and future work.

## 2 Multi-objective Job Shop Scheduling

Job shop scheduling problems are common and have been studied for over 50 years with practical applications in industrial context. In particular, the unrelated parallel machine scheduling problem represents a challenge in job shop scheduling, involving the allocation of $N$ jobs across $M$ unrelated machines. Typically, the term span denotes the completion time of a machine, while makespan refers to the maximum completion time considering all machines [2]. While minimizing makespan is considered the main and single objective, other significant

and conflicting objectives, such as average and standard deviation of the completion times are important [19]. Consequently, JSS problems are inherently multi-objective optimization problems, reflecting the real-world complexity of real world scheduling problems prevalent in industrial settings. This complexity highlights the need for trade-off solutions that capture a balance among various objectives.

### 2.1 Problem Description

In JSS problems, there are $N$ independent jobs that have to be processed in a set of $M$ available machines. Each machine $i$ can only execute one job $j$ at each instant for $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, N\}$. Moreover, all $N$ jobs do not have any precedence restriction and preemption is not allowed, that is, the processing of any job in a machine cannot be interrupted.

The following nomenclature will be considered: $p_{i,j}$ is the processing time for job $j$ on machine $i$, and $s_{i,j,k}$ is the setup time for job $j$ after job $k$ to be processed on machine $i$. The initial setup time for processing job $j$ on machine $i$ is denoted by $s_{i,j,0}$. The feasible sequence of all jobs assigned to machines is denoted by $\pi$. This vector includes all feasible subsequences $\pi_i$ on each machine $i$. The total completion time of jobs for the feasible sequence $\pi$ is represented by $c_i(\pi)$ for each machine $i$.

This work aims to find the sequence of jobs that simultaneously minimizes the makespan ($c_{max}(\pi)$), the average completion time ($\bar{c}(\pi)$) and the standard deviation of completion time ($c_\sigma(\pi)$). Mathematically, the objectives to be minimized can be expressed as follows:

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

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

$$c_\sigma(\pi) = \sqrt{\frac{\sum_{i=1}^{M}(c_i(\pi) - \bar{c}(\pi))^2}{M}} \tag{3}$$

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 jobs that precedes jobs $j$ in the subsequence $\pi_i$ of the feasible sequence $\pi$. It should be noted that when the job $j$ is the first job in the subsequence $\pi_i$, $k$ is zero and therefore $s_{i,j,0}$ is the setup time.

Therefore, these multi-objective JSS optimization problem with three objectives can be mathematical formulated as follows:

$$\min \mathbf{F}(\pi) = (f_1(\pi), f_2(\pi), f_3(\pi))^T = (c_{\max}(\pi), \bar{c}(\pi), c_\sigma(\pi))^T \tag{4}$$

where $\pi$ is a feasible sequence belonging to the feasible set $\Omega$ (the set of all possible permutations), and $\mathbf{F}(\pi)$ is the vector of objective functions defined in the objective space.

Since the objective space is a three-dimensional space, the comparison of solutions has to be based on the concept of Pareto dominance. Thus, for any two solutions $\pi_{\mathbf{a}}$ and $\pi_{\mathbf{b}}$ of the feasible set $\Omega$, a solution $\pi_{\mathbf{a}}$ is said to dominate a solution $\pi_{\mathbf{b}}$ ($\pi_{\mathbf{a}} \prec \pi_{\mathbf{b}}$) if $\forall_{i\in\{1,2,3\}} : f_i(\pi_{\mathbf{a}}) \leq f_i(\pi_{\mathbf{b}}) \wedge \exists_{j\in\{1,2,3\}} : f_j(\pi_{\mathbf{a}}) < f_j(\pi_{\mathbf{b}})$.

In this sense, since solutions are compared with respect to different criteria, there is no longer a single optimal solution, and a set of incomparable optimal solutions exists, generally known as the Pareto optimal set. The images of these Pareto optimal solutions define a Pareto front in the objective space, that allows the perception of trade-offs. Thus, the Pareto front is useful to assist the decision making process and can be expressed by $\mathbf{PS} = \{\pi_{\mathbf{x}} \in \Omega \mid \nexists \pi_{\mathbf{y}} \in \Omega : \pi_{\mathbf{y}} \prec \pi_{\mathbf{x}}\}$.

### 2.2 Multi-objective Approach

Multi-objective optimization algorithms are developed to deal simultaneously with problems with two or more objective functions. The objective space is in general complex since, usually, criteria conflict each other in such a way that when one is improved degrades some of the others. The goal of any multi-objective optimization algorithm is to find a set of solutions that represent trade-off between the objectives [15]. From an algorithmic perspective, designing an efficient optimization techniques is not simple, in particular, when the number of objectives is large. More recently, several algorithms have been developed to be efficient in dealing with these complex optimization problems with a large number of objectives, known as many-objective optimization problems [6].

NSGA-III [13] is an example of an evolutionary algorithm that based on the well-known NSGA-II [8] was extended and modified to tackle many-objective optimization problems. NSGA-III maintains the diversity of solutions along the search by using a set of reference points. The reference points are scattered on a hyperplane with a dimension to the number of objectives of the problem. Selection is based on the distances to these reference set and constitutes the main difference from its ancestor NSGA-II. During the search, each individual of the population is associated to the reference point with the closest perpendicular distance. If two individuals are associated to the same point, the individual with the shortest distance is chosen. This promotes diversity in population and convergence toward the Pareto front. Genetic operators (such as crossover and mutation) are used to generate new offspring from parent individuals and are similar to the ones used by NSGA-II.

Some features of the evolutionary algorithms are adequate to tackle combinatorial problems, since constraints can be handled by permutation representations and genetic operators can be specifically designed for this kind of search spaces.

## 3 Experimental Results

In order to study the NSGA-III performance, test problems were taken from previous works [2–4]. In this study, instances of JSS problems with balanced processing and setup time for $M \in \{2, 4, 6, 8\}$, and $N \in \{9, 10, 11\}$ were used.

There are 15 different instances for each combination of $M$ and $N$. The objectives were to minimize the makespan ($c_{max}(\pi)$), the average completion time ($\bar{c}(\pi)$) and the standard deviation of the completion time ($c_{\sigma}(\pi)$).

### 3.1 NSGA-III Implementation Details

The NSGA-III was used to solve the JSS optimization problem with three objectives. This algorithm was implemented in Python using the PYMOO framework [5]. A population size of 100 individuals was considered. The stopping criterion was based on the maximum number of generations and the period without improvement. The maximum number of generations was set at 10,000 and the period at 100 generations. Fourteen independent runs were made on the test problems because the algorithm used is stochastic.

Each chromosome of the population is a permutation of size $N + M - 1$ that expresses the feasible sequence of jobs on the machines, i.e., the sequence $\pi$. In the chromosome, genes larger than $N$ are used to divide the chromosome into $M$ subsequences ($\pi_i$) indicating the set of jobs and the corresponding order assigned to each machine $i$.

A binary tournament selection operator was used to select the chromosomes for being parents and generate offspring by the application of the genetic operators. The genetic operators were chosen to guarantee the feasibility of the solutions during the search: order-based crossover and inversion mutation. In the order-based crossover, the genes of two chromosomes are combined. So, two positions are generated at random and then the genes between those positions are swapped. The remaining empty positions are filled with the genes from the other parent, avoiding repeats. In the inversion mutation, two positions are generated at random and the genes between them are reversed.

### 3.2 Overall Analysis

The assessment of the performance of multi-objective optimization algorithms is not straightforward since it implies measuring its ability of convergence towards the Pareto optimal front as well as its ability to maintain diversity of solutions in objective space to facilitate the decision making process. So, in order to measure the performance of the NSGA-III, the hypervolume indicator ($H$) was used [21]. This measure allows to assess the algorithm performance both in terms of convergence to the Pareto front and diversity of the approximation along the Pareto front. All hypervolume indicator values were computed considering the reference point $(1000, 1000, 1000)^T$ and then standardized in order to belong to the interval $[0, 1]$. For a given problem, the higher the value of hypervolume, the best performance of the algorithm.

In Table 1, the statistics for the measured hypervolume for Pareto fronts obtained by NSGA-III for the 15 instances of the JSS problem for each combination of $M$ and $N$ are presented. For each instance of the JSS problem, 14 independent runs of NSGA-III were performed. Thus, for these 14 independent runs, the maximum (best) and mean hypervolume are computed and then the

respective average and standard deviation for each combination of $M$ and $N$. The average number of non dominated solutions (#ND) obtained for the best hypervolume of each instance and the mean number of function evaluations ($F_{\text{eval}}$) are also provided. In this table, $AVG(H_{\text{mean}})$ and $SD(H_{\text{mean}})$ are the average and standard deviation of the mean hypervolume (maximum hypervolume), and $AVG(H_{\text{max}})$, $SD(H_{\text{max}})$ are the average and standard deviation of the best hypervolume, respectively.

From this table, it can be observed that for the same value of $M$, the mean and maximum hypervolume decrease consistently when the value of $N$ increases. Moreover, for the same value of $M$, the standard deviations decreases as $N$ increases. This behavior is expected since scheduling a larger number of jobs to the same number of machines tends to be a more difficult problem. On the other hand, the mean number of non dominated solutions in the Pareto fronts (#ND) seems to be independent of $M$ and $N$. As expected, for the same $M$, the mean number of function evaluations ($F_{\text{eval}}$) grows with $N$ due to the higher difficulty of the problems.

**Table 1.** Statistics for the average of mean and maximum hypervolume, number of non dominated solutions and objective function evaluations.

| $M$ | $N$ | $AVG(H_{\text{mean}})$ | $SD(H_{\text{mean}})$ | $AVG(H_{\text{max}})$ | $SD(H_{\text{max}})$ | #ND | $F_{\text{eval}}$ |
|---|---|---|---|---|---|---|---|
| 2 | 9 | 0.1745 | 0.0156 | 0.1753 | 0.0154 | 7.8 | 14810.0 |
| | 10 | 0.1324 | 0.0144 | 0.1338 | 0.0145 | 4.3 | 17112.4 |
| | 11 | 0.0890 | 0.0120 | 0.0912 | 0.0120 | 7.0 | 17874.8 |
| 4 | 9 | 0.4655 | 0.0131 | 0.4688 | 0.0128 | 9.7 | 13798.6 |
| | 10 | 0.4348 | 0.0122 | 0.4390 | 0.0123 | 9.5 | 19267.6 |
| | 11 | 0.4092 | 0.0086 | 0.4174 | 0.0086 | 7.5 | 21959.8 |
| 6 | 9 | 0.6010 | 0.0112 | 0.6055 | 0.0116 | 10.7 | 19737.6 |
| | 10 | 0.5758 | 0.0087 | 0.5824 | 0.0080 | 9.3 | 22703.3 |
| | 11 | 0.5584 | 0.0063 | 0.5663 | 0.0068 | 7.3 | 25343.8 |
| 8 | 10 | 0.6350 | 0.0082 | 0.6406 | 0.0081 | 10.4 | 20049.0 |
| | 11 | 0.6158 | 0.0061 | 0.6222 | 0.0063 | 11.1 | 23621.4 |

In order to compare the performance of NSGA-III the data available on the Scheduling Research Virtual Center website[1] was used. This data provides information about the best makespan values known obtained by an improved Ant Colony Optimization (ACOII) [3] algorithm for the instances solved in this paper.

Table 2 presents, for each combination of $M$ and $N$ and each one of the 15 instances, the number of non dominated solutions found ($\#\text{ND}_{\text{total}}$) and the number of those solutions that have a makespan value equal or inferior to the best solution found by ACOII ($\#\text{ND}_{\leq c_{\max}}$). The last two columns of this table

[1] https://sites.wp.odu.edu/schedulingresearch/wp-content/uploads/sites/99/2016/01/Rm-Cmax-ACO-Arnaout-2014.xlsx.

indicate the number of instances in which $\#ND_{\leq c_{max}}$ is greater than 1 ($\#ND_{>1}$) and greater or equal than 1 ($\#ND_{\geq 1}$), respectively. All results presented in this table refer to the runs of NSGA-III with best hypervolume for each instance. The values of $\#ND_{>1}$ show that in the majority of the instances, more than one non dominated solution with the best value of makespan was found. This is important because it provides the decision maker with more than one alternative solution with the makespan value and differing on the average and standard deviation of the completion times in each machine. In addition, the values of $\#ND_{\geq 1}$ stress the advantage of using this approach since in the majority of the instances solutions with the best known makespan value were found. There are some instances in which NSGA-III was not able to find solutions with the best makespan value. However, NSGA-III found solutions close to those. Moreover, it should be noted that the results presented in this table were obtained with a number of function evaluations significantly inferior to the one used for ACOII [2,3]. Furthermore, NSGA-III was able to find a solution for the instance 3 of the combination $M = 2$ and $N = 11$ (signed in the table with *). This solution corresponds to the sequence $\pi =< \pi_1, \pi_2 >$ where $\pi_1 = [9, 10, 5, 4, 8, 6]$ and $\pi_2 = [3, 7, 11, 2, 1]$. The total completion times are $c_1(\pi) = 735$ and $c_2(\pi) = 738$, and therefore $c_{max}(\pi) = 738$, $\bar{c}(\pi) = 736.5$, and $c_\sigma(\pi) = 1.5$. For this instance 3, the best solution reported before has a makespan of 742 [3].

**Table 2.** Performance comparison of NSGA-III with the improved ant colony optimization (ACOII) algorithm.

| $M$ | $N$ | Instance | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | $\#ND_{>1}$ | $\#ND_{\geq 1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 9 | $\#ND_{total}$ | 8 | 10 | 4 | 8 | 7 | 7 | 10 | 7 | 10 | 6 | 9 | 3 | 10 | 7 | 11 | | |
| | | $\#ND_{\leq c_{max}}$ | 6 | 7 | 2 | 3 | 3 | 3 | 4 | 5 | 10 | 2 | 7 | 2 | 8 | 2 | 7 | 15 | 15 |
| | 10 | $\#ND_{total}$ | 5 | 3 | 4 | 6 | 2 | 4 | 4 | 7 | 4 | 7 | 2 | 3 | 5 | 4 | 4 | | |
| | | $\#ND_{\leq c_{max}}$ | 1 | 2 | 1 | 1 | 1 | 3 | 1 | 1 | 1 | 6 | 1 | 2 | 1 | 0 | 1 | 4 | 14 |
| | 11 | $\#ND_{total}$ | 5 | 5 | 6 | 9 | 7 | 9 | 3 | 6 | 6 | 8 | 5 | 11 | 6 | 8 | 11 | | |
| | | $\#ND_{\leq c_{max}}$ | 3 | 1 | 4* | 3 | 1 | 7 | 1 | 2 | 0 | 4 | 2 | 9 | 1 | 8 | 11 | 10 | 14 |
| 4 | 9 | $\#ND_{total}$ | 12 | 9 | 10 | 10 | 12 | 8 | 8 | 9 | 12 | 10 | 8 | 6 | 11 | 11 | 9 | | |
| | | $\#ND_{\leq c_{max}}$ | 4 | 4 | 3 | 3 | 2 | 7 | 4 | 5 | 2 | 5 | 2 | 5 | 4 | 2 | 4 | 15 | 15 |
| | 10 | $\#ND_{total}$ | 7 | 5 | 11 | 16 | 17 | 7 | 4 | 4 | 11 | 9 | 11 | 14 | 10 | 10 | 7 | | |
| | | $\#ND_{\leq c_{max}}$ | 5 | 2 | 2 | 2 | 1 | 3 | 0 | 2 | 7 | 2 | 2 | 4 | 3 | 3 | 4 | 13 | 14 |
| | 11 | $\#ND_{total}$ | 6 | 8 | 7 | 8 | 8 | 6 | 7 | 7 | 7 | 9 | 11 | 7 | 7 | 8 | 7 | | |
| | | $\#ND_{\leq c_{max}}$ | 2 | 0 | 0 | 2 | 2 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 3 | 8 |
| 6 | 9 | $\#ND_{total}$ | 13 | 7 | 8 | 7 | 7 | 11 | 5 | 11 | 13 | 16 | 24 | 11 | 8 | 7 | 12 | | |
| | | $\#ND_{\leq c_{max}}$ | 3 | 2 | 3 | 1 | 3 | 1 | 2 | 2 | 1 | 1 | 1 | 0 | 5 | 2 | 4 | 9 | 14 |
| | 10 | $\#ND_{total}$ | 5 | 8 | 7 | 11 | 14 | 13 | 13 | 10 | 5 | 11 | 10 | 7 | 8 | 11 | 7 | | |
| | | $\#ND_{\leq c_{max}}$ | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 0 | 0 | 0 | 2 | 1 | 3 | 8 |
| | 11 | $\#ND_{total}$ | 7 | 7 | 7 | 6 | 6 | 10 | 8 | 4 | 7 | 10 | 9 | 8 | 8 | 5 | 7 | | |
| | | $\#ND_{\leq c_{max}}$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | | 0 | 2 | 0 | 0 | 2 | 6 |
| 8 | 10 | $\#ND_{total}$ | 8 | 8 | 10 | 10 | 8 | 20 | 14 | 8 | 6 | 12 | 14 | 10 | 9 | 12 | 7 | | |
| | | $\#ND_{\leq c_{max}}$ | 4 | 4 | 2 | 3 | 3 | 4 | 1 | 0 | 2 | 3 | 2 | 1 | 4 | 2 | 2 | 12 | 14 |
| | 11 | $\#ND_{total}$ | 11 | 14 | 13 | 4 | 13 | 10 | 12 | 16 | 6 | 10 | 16 | 10 | 9 | 13 | 10 | | |
| | | $\#ND_{\leq c_{max}}$ | 1 | 3 | 4 | 2 | 3 | 1 | 3 | 2 | 0 | 0 | 0 | 4 | 4 | 1 | 3 | 8 | 12 |

### 3.3 Analysis of the Results of an Instance

In Fig. 1, the approximation to the Pareto optimal front obtained for the minimization of the makespan ($c_{\max}$), the average completion time ($\bar{c}$) and the standard deviation of the completion times ($c_\sigma$) for 4 machines and 9 jobs (instance 8) is depicted. In this figure, the makespan ($c_{\max}$) is represented in X-axis, the average completion time ($\bar{c}$) in the Y-axis, and the standard deviation of the completion times ($c_\sigma$) is represented by the size of the points.

For this instance, 9 different non dominated solutions were obtained by NSGA-III to approximate the Pareto optimal front. The trade-off represented by those solutions in terms of the 3 objectives can be clearly visualized in this 2D plot. This figure also includes the pay-off table indicating the values of the objectives for each solution. It can seen that there are 5 solutions with the best value of makespan ($c_{\max} = 328$) but with different values of the other objectives. These solutions are of great interest for the decision maker.

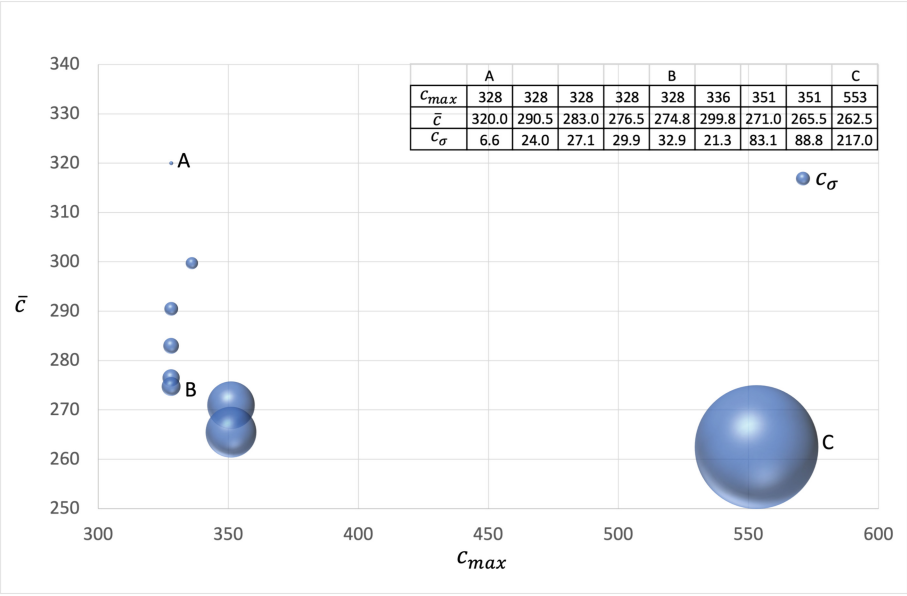



**Fig. 1.** Representation of the approximation to the Pareto optimal front for the minimization of makespan, the average completion time and the standard deviation of completion times for 4 machines and 9 jobs (instance 8).

Taking into account this approximation to the Pareto optimal front, 3 solutions (solutions A, B and C) were chosen as representative solutions to be analysed with more detail:

– Solution A is the solution with the best value of $c_{\max}$ and $c_\sigma$, and the worst value of $\bar{c}$;

- Solution B is the solution with the best value of $c_{\max}$, and intermediate values of $\bar{c}$ and $c_\sigma$;
- Solution C is the solution with the worst value of $c_{\max}$ and $c_\sigma$, and the best value of $\bar{c}$.

Solutions A and C are extreme solutions located on the upper-left and lower-right extremes of the Pareto front, respectively. Solution B is located on the elbow of the Pareto front and seems to be a good trade-off between the objectives.

It should be noted that if the 3 objectives were not considered as objectives to be minimized simultaneously, some important solutions would not be found. The following scenarios illustrate the importance of considering these 3 objectives for JSS problems:

- Scenario 1 - minimization of $c_{\max}$: any of the 5 solutions with $c_{\max} = 328$ could be found;
- Scenario 2 - minimization of $c_{\max}$ and $\bar{c}$, simultaneously: only solution B, solution C, and the two solutions between them could be found;
- Scenario 3 - minimization of $c_{\max}$ and $c_\sigma$: only solution A, solution C, and the 2 solutions between solution B and solution C could be found.

In Scenario 1, it is not guaranteed to find the solutions A and C that are good trade-offs. In Scenario 2, it is impossible to find solution A that is an interesting solution. Finally, Scenario 3 does not allow to find solution B. Thus, it is clear the importance of considering the 3 objectives, simultaneously.

In order to analyse solutions A, B and C, the data regarding instance 8 for $M = 4$ and $N = 9$ follows. The processing times $p_{i,j}$ of the job $j$ on machine $i$ are given in the following matrix:

$$p = \begin{pmatrix} 57\ 53\ 77\ 65\ 78\ 73\ 74\ 52\ 86 \\ 97\ 85\ 81\ 97\ 66\ 64\ 69\ 65\ 99 \\ 60\ 78\ 51\ 91\ 58\ 97\ 53\ 58\ 52 \\ 73\ 55\ 92\ 54\ 56\ 79\ 68\ 89\ 82 \end{pmatrix}.$$

The setup times $s_{i,j,k}$ to process job $j$ after job $k$ are given on matrices $s_1$, $s_2$, $s_3$, and $s_4$ for machines $m_1$, $m_2$, $m_3$, and $m_4$, respectively:

$$s_1 = \begin{pmatrix} 71\ 83\ 79\ 62\ 99\ 71\ 58\ 59\ 87 \\ 64\ 98\ 78\ 62\ 57\ 52\ 58\ 68\ 51 \\ 71\ 59\ 84\ 96\ 93\ 87\ 72\ 65\ 77 \\ 56\ 68\ 63\ 72\ 62\ 63\ 98\ 90\ 94 \\ 53\ 84\ 91\ 72\ 52\ 70\ 90\ 51\ 77 \\ 66\ 75\ 87\ 53\ 84\ 77\ 86\ 77\ 76 \\ 78\ 71\ 96\ 88\ 60\ 90\ 65\ 59\ 59 \\ 80\ 82\ 99\ 53\ 73\ 89\ 80\ 96\ 81 \\ 99\ 85\ 97\ 66\ 58\ 57\ 95\ 73\ 98 \end{pmatrix} \quad s_2 = \begin{pmatrix} 83\ 82\ 98\ 84\ 76\ 68\ 89\ 58\ 66 \\ 72\ 51\ 81\ 50\ 75\ 60\ 66\ 63\ 64 \\ 82\ 97\ 60\ 63\ 66\ 60\ 89\ 85\ 83 \\ 95\ 85\ 93\ 64\ 92\ 84\ 53\ 95\ 63 \\ 52\ 60\ 61\ 94\ 66\ 99\ 90\ 51\ 93 \\ 85\ 96\ 67\ 79\ 54\ 73\ 92\ 88\ 77 \\ 50\ 68\ 52\ 72\ 84\ 59\ 55\ 59\ 79 \\ 68\ 78\ 55\ 93\ 51\ 77\ 55\ 87\ 51 \\ 60\ 65\ 69\ 62\ 96\ 80\ 64\ 93\ 63 \end{pmatrix}.$$

$$s_3 = \begin{pmatrix} 60 & 55 & 59 & 81 & 87 & 61 & 68 & 95 & 73 \\ 67 & 99 & 96 & 53 & 93 & 79 & 69 & 90 & 60 \\ 81 & 73 & 58 & 77 & 77 & 90 & 55 & 58 & 80 \\ 51 & 77 & 70 & 85 & 65 & 56 & 56 & 71 & 59 \\ 52 & 59 & 54 & 78 & 82 & 50 & 74 & 59 & 96 \\ 78 & 88 & 75 & 70 & 70 & 54 & 85 & 73 & 64 \\ 89 & 76 & 94 & 97 & 95 & 77 & 52 & 90 & 59 \\ 57 & 50 & 93 & 86 & 77 & 63 & 97 & 51 & 62 \\ 52 & 63 & 92 & 85 & 61 & 64 & 72 & 99 & 56 \end{pmatrix} \quad s_4 = \begin{pmatrix} 50 & 90 & 68 & 77 & 85 & 67 & 85 & 80 & 86 \\ 98 & 97 & 53 & 96 & 50 & 88 & 71 & 79 & 98 \\ 98 & 74 & 90 & 53 & 53 & 89 & 71 & 53 & 63 \\ 62 & 95 & 77 & 63 & 99 & 89 & 69 & 95 & 69 \\ 53 & 67 & 83 & 81 & 86 & 81 & 86 & 84 & 56 \\ 82 & 81 & 91 & 68 & 70 & 77 & 65 & 99 & 66 \\ 87 & 93 & 67 & 97 & 88 & 55 & 63 & 70 & 68 \\ 71 & 99 & 62 & 60 & 59 & 82 & 70 & 54 & 91 \\ 76 & 89 & 94 & 75 & 78 & 79 & 60 & 80 & 87 \end{pmatrix}.$$

In the diagonal of the matrices $s_1$, $s_2$, $s_3$ and $s_4$, the setup time to process job $j$ at the initial instant on machines ($s_{i,j,0}$) are given.

For illustration purposes, Fig. 2 to Fig. 4 represent solutions A, B and C for instance 8 with 4 machines ($m_1$, $m_2$, $m_3$, and $m_4$) and 9 jobs ($n_1$, $n_2$, $n_3$, $n_4$, $n_5$, $n_6$, $n_7$, $n_8$, and $n_9$). In these figures, grey boxes represent the jobs. Additionally, the processing times and setup times are also indicated below the grey boxes and before the jobs, respectively.

Figure 2 represents the solution A. This solution corresponds to the sequence $\pi =< \pi_1, \pi_2, \pi_3, \pi_4 >$ where $\pi_1 = [8, 6]$, $\pi_2 = [4, 5]$, $\pi_3 = [3, 7, 9]$, $\pi_4 = [2, 1]$. The total completion times are $c_1(\pi) = 310$, $c_2(\pi) = 319$, $c_3(\pi) = 328$ and $c_4(\pi) = 323$, and therefore $c_{\max}(\pi) = 328$, $\bar{c}(\pi) = 320$, and $c_\sigma(\pi) = 6.6$. This solution is very good in terms of the makespan and since the standard deviation of the completion times is very low, the workload distribution is very balanced between machines. However, this solution is the worst in terms the average completion time.

In Fig. 3, solution B is represented. This solution corresponds to the sequence $\pi =< \pi_1, \pi_2, \pi_3, \pi_4 >$ where $\pi_1 = [1, 8]$, $\pi_2 = [2, 6]$, $\pi_3 = [3, 7, 9]$, $\pi_4 = [4, 5]$. The total completion times are $c_1(\pi) = 239$, $c_2(\pi) = 260$, $c_3(\pi) = 328$ and $c_4(\pi) = 272$, and therefore $c_{\max}(\pi) = 328$, $\bar{c}(\pi) = 274.8$, and $c_\sigma(\pi) = 32.9$. This solution B has the same makespan value of solution A. However, it is clear that the workload is not so well-distributed between the machines. When compared to solution A, the average completion time is significantly inferior.

In Fig. 4, the solution C is shown. This solution corresponds to the sequence $\pi =< \pi_1, \pi_2, \pi_3, \pi_4 >$ where $\pi_1 = [\,]$, $\pi_2 = [2, 6, 5]$, $\pi_3 = [8, 9, 1, 3, 7]$, $\pi_4 = [4]$. The total completion times are $c_1(\pi) = 0$, $c_2(\pi) = 380$, $c_3(\pi) = 553$ and $c_4(\pi) = 117$, and therefore $c_{\max}(\pi) = 553$, $\bar{c}(\pi) = 262.5$, and $c_\sigma(\pi) = 217.0$. In this solution, no jobs are assigned to machine $m_1$ and the workload is clearly unbalanced. In spite of being the solution with the lowest average completion time, this solution does not represent a good scheduling of the jobs.

Overall, solution A and solution B seem to be good trade-offs. Note that solution A and solution B share some similar features. For instance, the subsequence $\pi_3$ is the same for both solutions, meaning that the assignment of jobs on machine $m_3$ is the same. Thus, choosing between them depends on the preferences of the decision maker.

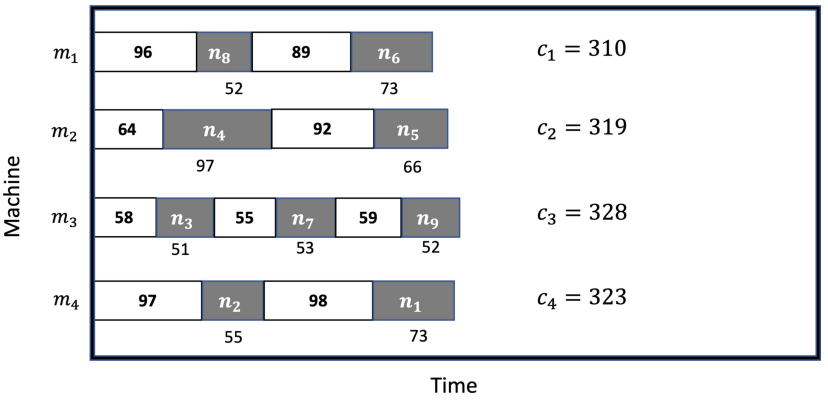



**Fig. 2.** Representation of solution A with 4 machines and 9 jobs (instance 8).

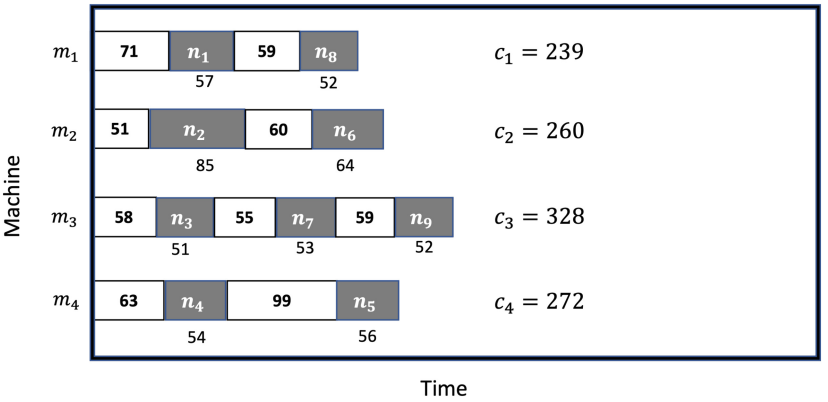



**Fig. 3.** Representation of solution B with 4 machines and 9 jobs (instance 8).

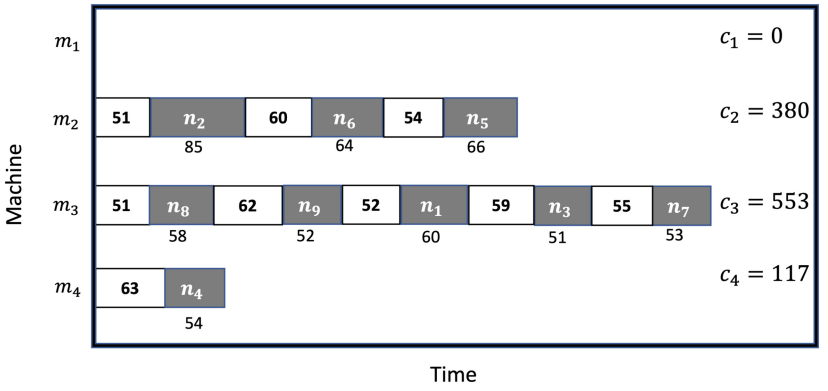



**Fig. 4.** Representation of solution C with 4 machines and 9 jobs (instance 8).

## 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 multi-objective optimization problems using NSGA-III. The makespan ($c_{\max}$), the average completion time ($\bar{c}$) and the standard deviation of the completion times were the objectives considered. In this work, 15 instances of JSS problems with balanced processing and setup times for $M$ machines ($M \in \{2, 4, 6, 8\}$), and $N$ jobs ($N \in \{9, 10, 11\}$) were used.

The hypervolume indicator was used to analyse the overall NSGA-III performance. These results indicate a good performance of NSGA-III to find approximations to the Pareto optimal set of JSS problems. The performance of NSGA-III was also compared with an improved Ant Colony Optimization (ACOII). These results demonstrate the advantage of using this approach since in the majority of the instances solutions with the best known makespan value were found. For some instances, the NSGA-III was not able to find solutions with the best makespan value, but solutions very close to those were found. However, it should be noted that in this work the number of function evaluations was significantly inferior to other approaches. Moreover, NSGA-III was able to find a solution for an instance that is better than the best solution reported before.

The analysis of some representative solutions from the approximation to the Pareto optimal set allows to confirm the validity and usefulness of this multi-objective optimization approach. Thus, the results also stress the importance of considering these three objectives, simultaneously. Also, the representation of the approximation to the Pareto optimal front highlights the trade-offs facilitating the decision making process.

Future work will include solving problems with more objectives and higher dimension, and to extend this approach to the distributed job shop problem. The performance of NSGA-III with other genetic operators for permutation representations will be also studied.

## 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. Arnaout, J.P., Rabadi, G., Musa, R.: A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. J. Intell. Manuf. **21**, 693–701 (2010)
5. Blank, J., Deb, K.: Pymoo: multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)

6. Chaudhari, P., Thakur, A.K., Kumar, R., Banerjee, N., Kumar, A.: Comparison of NSGA-III with NSGA-II for multi objective optimization of adiabatic styrene reactor. Mater. Today Proc. **57**, 1509–1514 (2022)
7. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2014)
8. 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)
9. Feldman, A.: Scheduling algorithms and systems. Columbia University (1999)
10. González-Neira, E.M., et al.: Robust solutions in multi-objective stochastic permutation flow shop problem. Comput. Ind. Eng. **137**, 106026 (2019)
11. Ibrahim, A., Rahnamayan, S., Martin, M.V., Deb, K.: EliteNSGA-III: an improved evolutionary many-objective optimization algorithm. In: 2016 IEEE Congress on evolutionary computation (CEC), pp. 973–982. IEEE (2016)
12. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools Appl. **80**, 8091–8126 (2021)
13. 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)
14. Liaqait, R.A., Hamid, S., Warsi, S.S., Khalid, A.: A critical analysis of job shop scheduling in context of industry 4.0. Sustainability **13**(14), 7684 (2021)
15. Para, J., Del Ser, J., Nebro, A.J.: Energy-aware multi-objective job shop scheduling optimization with metaheuristics in manufacturing industries: a critical survey, results, and perspectives. Appl. Sci. **12**(3), 1491 (2022)
16. dos Santos, F., Costa, L., Varela, L.: Multi-objective optimization of the job shop scheduling problem on unrelated parallel machines with sequence-dependent setup times. In: Gervasi, O., et al. (eds.) International Conference on Computational Science and Its Applications, pp. 495–507. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37108-0_32
17. 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.) Computational Science and Its Applications–ICCSA 2022 Workshops: Malaga, Spain, 4–7 July 2022, Proceedings, Part II, pp. 157–173. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10562-3_12
18. dos Santos, F., Costa, L.A., Varela, L.: Performance comparison of NSGA-II and NSGA-III on bi-objective job shop scheduling problems. In: Pereira, A.I., Mendes, A., Fernandes, F.P., Pacheco, M.F., Coelho, J.P., Lima, J. (eds.) International Conference on Optimization, Learning Algorithms and Applications, pp. 531–543. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-53025-8_36
19. Yenisey, M.M., Yagmahan, B.: Multi-objective permutation flow shop scheduling problem: literature review, classification and current trends. Omega **45**, 119–135 (2014)
20. 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)
21. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)