mathematics

MDPI

Article

A Statistical Comparison of Metaheuristics for Unrelated Parallel Machine Scheduling Problems with Setup Times

Ana Rita Antunes, Marina A. Matos, Ana Maria A. C. Rocha *, Lino A. Costa and Leonilde R. Varela

ALGORITMI Center, University of Minho, 4710-057 Braga, Portugal; b8769@algoritmi.uminho.pt (A.R.A.); mmatos@algoritmi.uminho.pt (M.A.M.); lac@dps.uminho.pt (L.A.C.); leonilde@dps.uminho.pt (L.R.V.)
* Correspondence: arocha@dps.uminho.pt

Abstract: Manufacturing scheduling aims to optimize one or more performance measures by allocating a set of resources to a set of jobs or tasks over a given period of time. It is an area that considers a very important decision-making process for manufacturing and production systems. In this paper, the unrelated parallel machine scheduling problem with machine-dependent and job-sequence-dependent setup times is addressed. This problem involves the scheduling of tasks on unrelated machines with setup times in order to minimize the makespan. The genetic algorithm is used to solve small and large instances of this problem when processing and setup times are balanced (Balanced problems), when processing times are dominant (Dominant P problems), and when setup times are dominant (Dominant S problems). For small instances, most of the values achieved the optimal makespan value, and, when compared to the metaheuristic ant colony optimization (ACOII) algorithm referred to in the literature, it was found that there were no significant differences between the two methods. However, in terms of large instances, there were significant differences between the optimal makespan obtained by the two methods, revealing overall better performance by the genetic algorithm for Dominant S and Dominant P problems.

Keywords: scheduling; unrelated parallel machines; sequence-dependent tasks; makespan; metaheuristics; genetic algorithm; statistical analysis

MSC: 90B36; 90C59; 90C27; 62P30; 68M20; 68Q25

Citation: Antunes, A.R.; Matos, M.A.; Rocha, A.M.A.C.; Costa, L.A.; Varela, L.R. A Statistical Comparison of Metaheuristics for Unrelated Parallel Machine Scheduling Problems with Setup Times. *Mathematics* **2022**, *10*, 2431. https://doi.org/10.3390/math10142431

Academic Editor: Ioannis G. Tsoulos

Received: 13 June 2022
Accepted: 7 July 2022
Published: 12 July 2022

1. Introduction

In the current 4th Industrial Revolution scenario, and with the underlying gradual transition of the use of exponential technologies and high-performance computing, companies, namely industrial ones, must be aware of the need to be able to progressively update their decision support tools, for instance, regarding manufacturing scheduling decision-making.

Thus, for an industrial company to remain successful and competitive, it is necessary to use effective and efficient methods and optimization algorithms, along with optimized production processes, in order to differentiate itself in the current global market. In this sense, the manufacturing scheduling decision support systems and underlying algorithms continue to play a crucial role by enabling manufacturing companies to obtain best-suited manufacturing schedules while avoiding unnecessary order cancellations or delays as well as the best use of production resources and costs; this has been a major concern of many researchers over the last decades.

Production management in an industrial environment faces several challenges due to dynamic changes in production and market volatility. Companies are increasingly looking to shorten delivery times as an economic measure due to increased competitiveness in emerging markets [1]. Industrial problems such as scheduling jobs or tasks are affected by customer requirements due to the variety of orders and speed of delivery [1,2].

Manufacturing scheduling is, therefore, a well-studied subject area and is considered a very important decision-making process for manufacturing and production systems. It aims to optimize one or more performance measures by allocating a set of resources to a set of jobs or tasks over a certain period of time [2].

Scheduling problems occur in different kinds of manufacturing environments, varying from a single stage of operation to multi-stage ones. In parallel machine scheduling problems, there is a need to assign tasks to machines coupled with sequencing problems. The scheduling problems in parallel machine environments have been the object of many studies in the last decades. However, cases on unrelated parallel machines are less investigated, especially when setup times are used [3–7].

Parallel machines are considered unrelated when the processing times of tasks depend on the machines to which they are assigned and when there is no relationship between the speeds of the machines [5]. The setup times are dependent since their values depend on both the work sequence and the configuration time matrix of each machine.

Many authors have considered the use of algorithms, heuristics, and metaheuristics to solve scheduling problems on unrelated parallel machines [3,6,8,9], although not all proposed approaches and underlying algorithms are equally effective and efficient in providing scheduling solutions. Yang et al. in 2022 [10] proposed an elite learning differential particle swarm optimization (DELPSO) in order to improve particle swarm optimization (PSO) for large problems. For this, two examples of guidance were applied to direct the update of each particle. The authors verified that the proposed model obtained better results when compared to the PSO. In another paper [11], a new approach was applied in order to improve the performance of multi-objective particle swarm optimizers (MOPSOs). The strategy used is called hybrid global leader selection (HGLSS), where Pareto dominance and density estimation are analyzed to verify the effectiveness of the proposed model. For this, the performance of the proposed approach was compared with nine multi-objective metaheuristics in solving several benchmark problems. The results showed an overcoming of the proposed algorithm in terms of the modified inverted generational distance indicator. The authors, Das and Suganthan [12], presented a literature review on the evolutionary algorithm (EA). In addition, an overview of works carried out on differential evolution (DE) was conducted, where subjects such as variants, multi-objective applications, constrained optimization problems, and theoretical studies of DE were addressed. Finally, engineering applications in which DE was applied were presented. In Pan et al. [13], a differential evolution (DE) algorithm was presented to solve a permutation flow shop scheduling problem in order to minimize the makespan. DE is a traditional continuous algorithm, and the lowest position value rule was presented in order to convert the continuous vector to a discrete work permutation.

This paper focuses on single-stage scheduling problems occurring in parallel machine environments. It is intended to evaluate the behavior of the genetic algorithm (GA) when applied to the scheduling problem of unrelated parallel machines in order to minimize the makespan of a set of tasks subject to varying setup times. A set of small and large instances of this problem will be used to assess the GA performance when compared to the solutions obtained by the metaheuristic ant colony optimization (ACOII) [8,9]. Then, statistical analysis of these two metaheuristics is performed.

In order to properly put forward the main contributions of this work, this paper is organized as follows. In the second section, a summarized introduction to manufacturing scheduling is provided. In Section 3, the unrelated parallel machine scheduling problem is briefly described. Section 4 presents the guidelines for the computational study to be carried out, the metaheuristic GA, and some implementation details. In Section 5, a comparative statistical analysis of the metaheuristics GA and ACOII is performed for small and large instances of scheduling problems, and the main conclusions and planned future work are summarized in Section 6.

2. Manufacturing Scheduling

2.1. General Overview

Scheduling plays a very important role in the decision-making process to be carried out in different decision-making environments, from manufacturing to information processing, transportation, and distribution, among other kinds of services [2]. In manufacturing, scheduling problems vary from single-stage or operations problems, occurring in single or parallel machine environments, up to multi-stage or operation ones, typically occurring in flow shop (FSP), job shop (JSP), or open shop scheduling problems (OSPs), to mention some of the most well-known [2]. The multi-stage scheduling problems tend to be more complex than the single-stage ones; for instance, FSPs have some complicated variants, namely, considering energy issues and material handling assumptions or stochastic data [14,15].

Manufacturing scheduling implies the allocation of jobs or tasks to resources or vice versa, its sequencing or order in time for being processed, along with the definition of the initial and final processing times for each job or task in a given resource, made available through a given manufacturing environment [16].

In [17], Brucker points out that the scheduling of a production system is dependent on several factors, such as manufacturing orders, available resources, available machines, and the operations to be performed by each one, and there may be processing times that are different from one machine to another one while satisfying and/or optimizing a single or complex performance measure or criteria.

The programming of parallel machines has been a growing research area since the first works carried out in the early 20th century [18]. This type of problem has, since then, received continuous interest from researchers due to its relevance to manufacturing environments [3,7–9].

Manufacturing scheduling to optimize configurations, either directly or indirectly, has been an important issue for different types of industries, including plastic, textile, and chemical industries, as well as for some service areas [3,7,9,19–25].

Allahverdi, in [26], presents a review of scheduling problems. In his work, the scheduling problems are further classified based on the underlying production environment as a single machine, parallel machines, flow shop, job shop, or open shop. It also classifies them according to the consideration and processing of information regarding their inclusion in family sets, besides the characterization of sequence-dependent jobs/tasks or machine configurations, which also affects production times and/or costs, among other characterization parameters that are also further explored in other publications, such as [3,21,22].

Thus, scheduling requires decisions about jobs/tasks and processing resources. The sequencing corresponds to a permutation of jobs/tasks or the order in which they might be processed on each resource or machine. On the other hand, the allocation of resources or machines refers to the choice of which one will process each job or task [27]. The scheduling problem aims to assign tasks to machines and define the periods that each task is processed on each machine in order to minimize and/or maximize an optimization criterion, usually expressed in the form of an objective function intended to be optimized [11].

The Brucker classification system, in [17], uses the nomenclature $\alpha \mid \beta \mid \gamma$, initially introduced by [28]. In this nomenclature, the α represents the scheduling classification factors related to the manufacturing environment, which usually include the type of manufacturing system and the number of machines. The manufacturing environment can range from a simple one, such as a single machine, up to more complex ones, occurring in flow shop, job shop, and open shop environments or flexible manufacturing environments, besides other kinds of manufacturing systems, for instance, taking place in different types of parallel machine environments. These manufacturing environments can have different complexity levels, not just according to the nature of the manufacturing systems' configuration and the underlying production flows themselves but further to other, additional characteristics, about a more or less widened set of conditions and constraints imposed in the scheduling problem, expressed by a corresponding β set of factors in the problem classification nomenclature. Moreover, a simple or combined set of performance measures can also be

considered and expressed through the γ factor. Since its introduction, this notation has been used and reformulated by several authors, and many classifications have been added as other problems arise [22].

2.2. Scheduling Assumptions

The scheduling problem studied in this work occurs in a parallel-machines scheduling environment, where the processing times of the task are expressed through a task-machine tuple, which varies from one machine to another for processing a given task [29,30].

In scheduling problems arising in manufacturing environments, frequently, multiple machines are available for processing a set of tasks, and the processing times are often even more dependent on task sequences. Thus, there is a sequence-dependent setup time ($S_{i,j,k}$) whenever, after processing task j, preparation time $S_{i,j,k}$ is required before processing task k. Moreover, when these times are also dependent on the machines, index i is added [31–33].

The scheduling problem considers a set of tasks with setup times that are dependent on the machine used and sequence-dependent on the unrelated parallel machines set, with the goal of minimizing the maximum completion time or makespan. In scheduling theory, the makespan (C_{max}) is defined as the completion time of the final task of a job to be processed (when the job leaves the system). The scheduling problem is thus based on a set of N tasks that must be processed on a machine from a set of M unrelated parallel machines (R_M). The processing time of the tasks depends on the assigned machine, and there is no relationship between these machines as they are unrelated. The setup times are machine- and task-sequence-dependent ($S_{i,j,k}$). Each machine has its setup time matrix, and each matrix is different from the others for the remaining unrelated parallel machines.

The problem studied in this work is classified in the literature as $R_M \mid S_{i,j,k} \mid C_{max}$. Minimizing the makespan of a scheduling problem with identical parallel machines and sequence-dependent setup times is categorized as NP-hard. Therefore, the most complex problem of unrelated parallel machines is also considered NP-hard.

2.3. Review about Scheduling Sequence-Dependent Setups in Unrelated Parallel Machines

The scheduling problem occurring in unrelated parallel machines for processing sequence-dependent setup times is quite important as it can be found in several areas such as the electronics, steel, and textile industries [34–37].

Kim et al. [35] used the simulated annealing (SA) algorithm to solve a scheduling problem in the electronics industry. In their work, it was possible to conclude that the proposed SA method significantly outperformed a neighborhood search method regarding the total delay of jobs or tasks.

Tang and Wang [36] formulated a scheduling problem for the steel industry as a mixed nonlinear program and proposed the Tabu search (TS) heuristic to obtain satisfactory solutions. The results showed that their model and heuristic performed more efficiently and effectively than other manual planning approaches.

In the textile industry, Gendreau, Laporte, and Guimarães [34] applied a heuristic to the multiprocessor scheduling problem with sequence-dependent setup times; their results showed that their heuristic was faster than a TS-based one but, at the same time, provided solutions of almost similar quality.

Thus, the complexity of the scheduling problem of unrelated parallel machines has led to increased interest in heuristic procedures to find solutions in a reasonable time interval. Kim and Chen [38] proposed four research heuristics for the aforementioned problem. According to the authors, these heuristics can be easily applied to obtain practical production scheduling solutions. Ghirardi and Potts [39] also studied the problem of unrelated parallel machines for minimizing the makespan; the underlying heuristic used was an application of the recovering beam search technique. The computational results allowed them to generate approximate solutions for large instances of problems (up to 50 machines and 1000 jobs) in just a few minutes.

Another heuristic, called Meta-RaPS, was introduced by Rabadi, Moraga, and Al-Salem [7] to minimize the makespan in unrelated parallel machine problems with sequence-dependent setup times. The performance of the proposed heuristic was evaluated by comparing its solutions with those obtained by other existing heuristics for the same problem. The results showed that the Meta-RaPS found optimal solutions for small problems and performed better than other existing heuristics for larger problems.

For the same problem and makespan objective function, Arnaout et al. [8] introduced the ant colony optimization (ACO) approach. To evaluate the performance of the ACO, the authors compared their solutions with the ones obtained by other heuristics, for example, solutions based on Tabu search and a partitioning heuristic with those obtained by Meta-RaPS [7]. They concluded that the ACO outperformed the other algorithms.

Arnaout et al. [9] also proposed an improved ACO algorithm (ACOII) and mentioned achieving better performance than the previous version; further, the algorithm had the ability to solve more difficult combinatorial optimization problems by partitioning them into subproblems.

The minimization of the makespan is one of the most studied criterion in the production scheduling literature, whether in parallel or single machines. For example, Woo, Jung, and Kim, in [23], developed a mixed-integer linear programming (MILP) model to find the optimal solution to the scheduling problem in unrelated parallel machines with the aim of minimizing the makespan. They proposed a new rule based on a genetic algorithm with a chromosome that represents the sequence of assignment of jobs or tasks to a machine, with the scheduling of jobs/tasks for each machine being determined by a heuristic based on completion time during the chromosome decoding process.

Considering that the setup times are dependent on the work sequence, the authors of [6] presented a GA for minimizing the makespan when solving scheduling problems in unrelated parallel machines. Their GA algorithm was further compared with other algorithms found in the literature, and they concluded that their proposed GA outperformed existing ones.

More recently, the scheduling of unrelated parallel machines for green manufacturing purposes, with resource constraints, was proposed by Zheng and Wang [40]. This work aimed at minimizing the makespan and total carbon emissions; to solve the problem, a collaborative multi-objective fruit fly optimization algorithm (CMFOA) was proposed. The results showed that their multi-objective algorithm was able to obtain more and better non-dominated solutions than other existing algorithms in comparison.

Aydilek et al. [41] addressed a scheduling problem to minimize order delays, in which the setup times were independent of the processing times, through the application of algorithms of self-adaptive differential evolution and hybrid and simulated insertion algorithms. A scheduling problem with different approaches to setup times, aiming to minimize the makespan with the application of an enhanced version of the ACO algorithm, was studied in [24,25].

Abreu and Prata [42] presented a hybrid GA for solving the unrelated parallel machine scheduling problem with sequence-dependent setup times. A case study on the granite industry is presented, and the proposed approach outperformed three traditional dispatch rules presented in the current literature. Gedik et al. in [43] studied the non-preemptive unrelated parallel machine scheduling problem with job/task-sequence- and machine-dependent setup times in order to minimize the makespan. Their study provided a novel constraint programming (CP) model with two customized branching strategies that used CP's global constraints, interval decision variables, and domain filtering algorithms. According to the authors, in terms of average solution quality, the computational results indicated that their CP model slightly outperformed their analyzed contributions from state-of-art algorithms in solving small problem instances and was able to prove the optimality of 283 currently best-known solutions. It is also mentioned to be effective in finding good quality feasible solutions for larger problem instances. Fanjul-Peyro et al. in [44] studied the same problem with the same objective function, but they modeled the

problem by means of two integer linear programming problems. One was based on a model previously proposed in the literature, and the other was based on packing problems. According to the authors, since their models were unable to solve medium-sized instances to optimality, they proposed three other metaheuristics for each of these two models. Their results showed that the proposed metaheuristics significantly outperformed the mathematical models. Recently, in [45], Arnaout addressed the unrelated parallel machine scheduling problem with setup times when minimizing the makespan through a worm optimization (WO) algorithm. The performance of the WO algorithm was evaluated by comparing its solutions to solutions of six other known metaheuristics.

Considering the previously presented literature review, most of the research addressed the scheduling problem with different objective functions and algorithm applications in different contexts and industrial environments.

The work underlying this paper was motivated by the work carried out by Arnaout et al. [9], where a comparison was made with the solutions obtained by the Meta-RaPS metaheuristic [46]. Based on data for small and large problem instances, this paper aims to propose a genetic algorithm for solving unrelated parallel machine scheduling problems with setup times for minimizing the makespan, C_{max}. Moreover, in this study, we intend to present an extended evaluation of the behavior of the GA when solving different types of case studies (small and large instances) of unrelated parallel machine scheduling problems with setup times.

3. The Scheduling Problem

This paper addresses the unrelated parallel machine scheduling problem considering the scheduling of N tasks that are available at time zero on M unrelated machines (R_M). The objective function is the makespan C_{max}, considering machine-dependent and sequence-dependent setup times $S_{i,j,k}$. This problem is classified in the literature as $R_M \mid S_{i,j,k} \mid C_{max}$, which is a generalization of the $P_M \mid \mid C_{max}$ problem of identical speeds for processing a set of tasks on the machines [2,47]. The unrelated parallel machine scheduling problem is known to be NP-hard [2] and can be formulated as a mixed-integer linear programming (MILP) model.

In the following, the problem assumptions are described:

- M is the number of parallel machines;
- N denotes the number of tasks to be scheduled;
- Each machine can only process one task at a time without preemption;
- For the initial time instant, which is at time zero, all tasks are available. No restrictions of precedence are imposed among tasks;
- For each machine i, each task j has a processing time, $p_{i,j}$;
- For each machine i, for processing tasks j just after tasks k, there is a setup time, $S_{i,j,k}$. The setup time is different for each machine;
- The objective is to minimize the makespan C_{max}. The term span is used to define the completion time of a machine, while the term makespan is used for the maximum span in the solution of the problem.

The mathematical programming model of the considered problem is presented below, which consists of finding an optimal solution to schedule a set of jobs or tasks in a set of unrelated parallel machines regarding the existence of sequence-dependent setup times, a similar model to the one used by Guinet in [8,48]. This MILP model includes binary variables ($x_{i,j,k} \in \{0,1\}$) indicating the assignment of tasks to machines and continuous variables denoting the completion times of tasks ($C_j \geq 0$ and $C_{max} \geq 0$).

$$Min\ C_{max} \tag{1}$$

subject to

$$\sum_{\substack{i=0 \\ i\neq j}}^{N}\sum_{k=1}^{M} x_{i,j,k} = 1, \qquad \forall j = 1,\dots,N \tag{2}$$

$$\sum_{\substack{i=0 \\ i\neq j}}^{N} x_{i,h,k} - \sum_{\substack{j=0 \\ j\neq h}}^{N} x_{h,j,k} = 0, \qquad \forall h = 1,\dots,N,\ \forall k = 1,\dots,M \tag{3}$$

$$C_j \geq C_i + \sum_{k=1}^{M} x_{i,j,k}\left(S_{i,j,k} + p_{j,k}\right) + HV\left(\sum_{k=1}^{M} x_{i,j,k} - 1\right), \qquad \forall i = 0,\dots,N,\ \forall j = 1,\dots,N \tag{4}$$

$$\sum_{j=0}^{N} x_{0,j,k} = 1, \qquad \forall k = 1,\dots,M \tag{5}$$

$$C_j \leq C_{max}, \qquad \forall j = 1,\dots,N \tag{6}$$

$$x_{i,j,k} \in \{0,1\}, \ \forall i = 1,\dots,N,\ \forall j = 1,\dots,N, \ \forall k = 1,\dots,M \tag{7}$$

$$C_0 = 0 \tag{8}$$

$$C_j \geq 0, \qquad \forall j = 1,\dots,N \tag{9}$$

where

C_j: maximum completion time of task j;
$p_{j,i}$: processing time of task j in machine i;
$S_{i,j,k}$: setup time dependent on the processing sequence of task j after task k in machine i;
$S_{0,j,i}$: setup time for processing first task j on machine i;
$x_{k,j,i}$: 1 if task j is processed immediately after task k in machine i, and 0 otherwise;
$x_{0,j,i}$: 1 if task j is the first one to be processed in machine i, and 0 otherwise;
$x_{j,0,i}$: 1 if task j is the last task to be processed in machine i, and 0 otherwise;
HV: a large positive number (usually denoted by a capital M).

The objective function (1) intends to minimize the makespan, where C_{max} is the length of time that elapses from the start of jobs to the end of the last job. Constraints (2) ensure that each task is only scheduled once and processed by a single machine. Constraints (3) ensure that each task must not be preceded or succeeded by more than one task. Constraint (4) calculates the completion time and ensures that no task precedes or succeeds the same task. Constraint (5) ensures that only one task can be scheduled first on each machine. There is no need for additional constraints to ensure that only one task is scheduled last on each machine because Constraints (5) and (3) guarantee this. Constraints (6) express the makespan C_{max} as a variable that must be larger than any other job's completion time. Constraints (7) guarantee that the decision variable x is binary in all domains. Constraint (8) states that the completion time for dummy work is zero, and Constraint (9) ensures that the completion time is non-negative. Solving the scheduling problem described above enables optimal solutions to be obtained.

4. Computational Study

This section presents the way that this computational study will be conducted. Firstly, the scheduling data description will be presented. Then, the genetic algorithm will be briefly described, followed by the implementation details considered to achieve the goals of the research.

4.1. Scheduling Data Description

The data available on the Scheduling Research Virtual Center page (available online: https://sites.wp.odu.edu/schedulingresearch/wp-content/uploads/sites/99/2016/01/Rm-Cmax-ACO-Arnaout-2014.xlsx (accessed on 10 May 2021) [49] were used for

comparison with the GA. This work is divided into two large groups: first, a comparative analysis of the solutions obtained by GA for small problems, and then, a statistical analysis of the solutions obtained with large problems.

The M and N values define the dimension of the problem in terms of the number of machines and tasks: small problems for some combinations of M in $\{2, 4, 6, 8\}$, and N in $\{6, 7, 8, 9, 10, 11\}$ were considered, and large problems were defined for some combinations of M in $\{2, 4, 6, 8, 10, 12\}$ and N in $\{20, 40, 60, 80, 100, 120\}$.

For each combination of the number of machines and tasks, there are 15 instances of problems. The study was carried out for three types of small and large problems: Balanced (when processing and setup times are balanced), Dominant P (when processing times are dominant), and Dominant S (when setup times are dominant).

4.2. Implementation Details

In this work, the implementation of the genetic algorithm [50–52] to solve the unrelated parallel machine scheduling problem with machine-dependent and job-sequence-dependent setup times is addressed. The *ga* function from MATLAB®, which implements the genetic algorithm, was used to solve this combinatorial problem. However, an implementation of a permutation to represent the solutions was previously defined in order to use the *ga* function. GA works with a population of chromosomes that represent the potential solution of the optimization problem and is adequate to tackle combinatorial problems since the constraints can be handled by permutation representations. For this scheduling problem, a solution is represented by a chromosome (a sequence of genes) that is a permutation of size $N + M - 1$. In this representation, a solution is a sequence of integer values that can occur only once. In a chromosome, the permutation values that are superior to N divide the chromosome into M subsequences that indicate the tasks and the corresponding order assigned to each machine. The genetic operators implemented to guarantee the feasibility of the solutions during the search were the order-based crossover and swap mutation. In the order-based crossover, the genetic material between two chromosomes is combined in order to generate offspring. Two random positions are selected, 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 swap mutation, two positions are generated at random, and the genes in a chromosome are swapped. A stochastic uniform selection operator was used to select chromosomes for the application of genetic operators.

The population size and the maximum number of generations of GA were set to 50 (default value) and 15,000, respectively. The crossover fraction was 0.8, i.e., each generation of the order-based crossover was applied to 80% of the population. The swap mutation was applied to the same fraction of the population. GA parameter values were chosen, taking into account the ACOII parameter values used in [9] in order to guarantee a fair performance comparison between the algorithms. Due to the stochastic nature of the GA, 30 independent runs were performed; the value of MaxStallGenerations was 1000. The case studies were run in the MatLab® R2021a using a 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80 GHz 2.80 GHz.

After obtaining the results using GA, the pandas and scipy.stats libraries [53,54] for Python version 3.8 were used to perform the statistical analysis.

Firstly, data were imported using the pandas' library by applying the *read_csv* function. Then, the function *groupby* was performed to identify the minimum value of the makespan, considering the same number of machines, tasks, and instances.

Thereafter, the library scipy.stats was used to implement the *shapiro*, *ttest_rel* and *wilcoxon* functions to evaluate if data followed a normal distribution and to apply a parametric *t*-test and non-parametric Wilcoxon test for the related samples, respectively.

In the case of data following a normal distribution, a *t*-test was considered to compare the metaheuristics GA and ACOII; otherwise, the Wilcoxon test was performed. After identifying the combinations of M and N values where there are differences between

the two methods, the confidence interval for paired samples was computed, taking into consideration Equation (10) [55].

$$\overline{d} - t_{(n-1;\frac{\alpha}{2})} \times \frac{s_{\overline{d}}}{\sqrt{n}} < \mu_d < \overline{d} + t_{(n-1;\frac{\alpha}{2})} \times \frac{s_{\overline{d}}}{\sqrt{n}} \tag{10}$$

First, the differences between the makespan of ACOII and GA must be performed, defined as d. Moreover, $\overline{d}$ and $s_{\overline{d}}$ are the mean and standard deviation values of d, respectively. Furthermore, t is a quantile from the t-Student distribution, where n is the number of observations and α is the level of significance.

In order to visualize the obtained results, the seaborn library [56] was implemented to display the boxplot graphs using the *boxplot* function. Boxplots allow us to compare the distribution of the makespan values obtained by each algorithm for the different combinations of M and N.

5. Comparative Statistical Analysis

In this section, comparative statistical analysis is performed considering the makespan and the execution time obtained by the metaheuristics GA and ACOII when solving small and large instances arising in the scheduling problem on unrelated parallel machines with sequence-dependent setup times.

5.1. Small Problems

The comparison between both algorithms, GA and ACOII, was made considering the performance measures given by Equations (11) and (12) to identify which method achieved better results for small and large problems.

$$\gamma = \frac{C_{max}(Algorithm) - C_{max}(Optimal)}{C_{max}(Optimal)} \tag{11}$$

$$\delta = \frac{C_{max}(ACOII) - C_{max}(GA)}{C_{max}(GA)} \tag{12}$$

For the small instances, the optimal values are known when $M = 2$ and $N \in \{6, 7, 8, 9\}$, $M = 4$ and $N \in \{6, 7, 8\}$, and $M = 6$ and $N = 8$. Note that if the GA method, in the small instances, achieves the optimal value, then γ must be equal to zero. If γ is positive, it means that the proposed method achieved a worse value (greater value in terms of minimization) than the optimal value; otherwise, a better value was found (smaller). In terms of δ values, if it is positive, then GA achieved better results than ACOII; otherwise, ACOII achieved better results than GA.

Table 1 presents the average values of γ for the small instances where the optimal value is known. According to the results, in most of them, GA found the optimal value, except for the Dominant S ($M = 4,\ N = 7$) and Dominant P problems ($M = 4, N = 6$). For this Dominant S problem, the γ value is positive ($\gamma = 0.0005$), which means the result achieved is close to the optimal value. Furthermore, in the Dominant P problem, the γ value is negative ($\gamma = -0.00034$); thus, a better result than the optimal known value is found.

The δ values for the two algorithms are presented in Table 2 for Balanced, Dominant S, and Dominant P problems. In the Balanced instances, there are three cases where GA achieved better results ($\delta > 0$) and one case where ACOII had a better result ($\delta < 0$). In terms of Dominant S instances, there are three cases where GA had better results than ACOII and two cases where ACOII achieved better results than GA. Finally, in the Dominant P instances, there are also three cases where GA had the best performance and one case where ACOII performed better than GA. Overall, the GA method achieved better results when compared to ACOII, although these are very small differences.

Table 1. Average γ deviation from optimal solutions for small problems.

M	N	Balanced		Dominant S		Dominant P	
		ACOII	GA	ACOII	GA	ACOII	GA
2	6	0	0	0	0	0	0
	7	0	0	0	0	0	0
	8	0	0	0	0	0	0
	9	0	0	0	0	0	0
4	6	0	0	0	0	0	−0.00034
	7	0	0	0	0.0005	0	0
	8	0	0	0	0	0	0
6	8	0	0	NA	NA	NA	NA

Table 2. Average δ deviation from GA for small problems.

M	N	Balanced	Dominant S	Dominant P
2	10	0	−0.00026	−0.00026
	11	0.000469	0	0
4	6	0	0	0.000336
	7	0	−0.0005	0
	9	0	0.000117	0.00282
	11	−0.00055	0.000112	0
6	10	0.000271	0	0
	11	0.001328	0	0.000167
8	11	0	0.000351	0

Hypothesis testing has been used to test if there are statistically significant differences between the two methods (GA and ACOII). Parametric tests were first considered. Therefore, the underlying assumptions were tested, namely, the normality of data. If so, then the parametric *t*-test for comparing the means of two related samples was computed. Otherwise, a non-parametric Wilcoxon test was used. The hypotheses to be tested are:

H_0: There are no differences between the GA and ACOII results.

H_1: There are differences between the GA and ACOII results.

Considering the level of significance as 5%, if the *p*-value is greater than 0.05, it means that the null hypothesis (H_0) is not rejected. Otherwise, it is rejected.

For the small instances, the hypothesis test was performed for *M* and *N* values that presented some differences between the GA and ACOII methods (see Table 2). The *p*-values and the 95% confidence intervals (CI) are shown in Table 3. According to the results achieved, all the *p*-values were greater than 0.05, considering a significance level of 5%. Thus, the null hypothesis (H_0) is not rejected, and consequently, there are no statistically significant differences between GA and ACOII results for small instances. The same conclusion can be drawn using the confidence interval since the value zero belongs to all confidence intervals. In other words, the mean difference between the average δ deviations obtained for GA and ACOII can be equal to zero.

Table 4 shows the average execution times, in seconds, for each type of small instance, where the time increases as the number of concurrent machines and the number of tasks increase. It is possible to observe that for the Dominant P problems, a higher average execution time is required when the number of machines is 8. On the contrary, for the Dominant S problems, the lowest average execution time is observed for the number of machines equal to 2.

Table 3. Hypothesis test *p*-values and 95% confidence intervals for small problems.

M	N	Balanced		Dominant S		Dominant P	
		p-Value	CI	*p*-Value	CI	*p*-Value	CI
2	10	-	-	0.334	[−0.305;0.839]	0.334	[−0.305;0.839]
	11	0.334	[−1.048;0.382]	-	-	-	-
4	6	-	-	-	-	0.334	[−0.419;0.153]
	7	-	-	0.334	[−0.229;0.629]	-	-
	9	-	-	0.334	[0.210;0.076]	0.334	[−5.032;1.832]
	11	0.334	[−0.229;0.629]	0.334	[−0.210;0.076]	-	-
6	10	0.317	[−0.210;0.076]	-	-	-	-
	11	0.461	[−1.078;0.412]	-	-	0.751	[−0.509;0.376]
8	11	-	-	0.334	[0.419;0.153]	-	-

Table 4. Average GA execution time for Balanced, Dominant P, and Dominant S for small problems.

			Number of Tasks (N)					
			6	7	8	9	10	11
Number of Machines (M)	Balanced	2	11.44	11.76	12.03	12.34	12.71	12.79
		4	13.60	14.16	14.38	14.33	14.85	15.02
		6	-	-	15.16	15.77	16.13	15.85
		8	-	-	-	-	16.20	16.63
	Dominant P	2	10.97	11.11	11.39	12.24	12.54	12.66
		4	13.64	14.15	13.79	13.67	14.00	14.42
		6	-	-	14.77	15.88	16.19	16.32
		8	-	-	-	-	17.13	17.61
	Dominant S	2	11.34	11.81	12.01	11.35	11.69	12.33
		4	12.99	13.89	14.29	14.46	14.77	14.95
		6	-	-	14.54	15.37	15.01	15.13
		8	-	-	-	-	16.54	16.83

5.2. Large Problems

The same analysis was reproduced for large problems to compare GA and ACOII performances. The computed values for the average makespan deviation from GA (δ) are presented in Table 5. For the Balanced instances, negative values were obtained for all instances, which means that the ACOII method has better performance than GA. However, GA has better results in Dominant P and Dominant S instances, even though ACOII presents better results in Dominant P instances when the number of machines is equal to 12 and in Dominant S instances when $M = 12$ and $N = 40$.

Parametric and non-parametric tests were applied to statistically prove if there are significant differences between the average δ deviations achieved by the GA and ACOII algorithms for large instances. The hypotheses to be tested were the same as the ones previously defined for small instances. In terms of results, all *p*-values were less than 0.05; therefore, for the large instances, the null hypothesis is rejected, and it is possible to conclude that there are significant differences between GA and ACOII results considering the significance level of 5%. Furthermore, it is also possible to conclude that there are significant differences between the two algorithms for the significance level of 0.1% since all *p*-values are less than 0.001.

The next step was to compute the mean and median C_{max} values for each algorithm to identify which one achieved better results. For example, considering $M = 12$ and $N = 40$, in Dominant S instances, the mean values were 748.33 and 737.60, and the median values were 749.00 and 739.00 for GA and ACOII, respectively. Thereby, in this case, ACOII performed better than GA since it achieved lower mean and median C_{max} values. This analysis was conducted for all combinations of M and N. Overall, for Balanced instances,

the ACOII results were significantly better than the GA results. On the other hand, for the Dominant P and Dominant S instances, the GA algorithm obtained better results than ACOII, except when $M = 12$ and $N = 40$ for Dominant S and $M = 12$ for Dominant P, where ACOII achieved better results.

Table 5. Average δ deviation from GA for large problems.

M	N	Balanced	Dominant S	Dominant P
2	20	−0.0052	0.6010	0.5968
	40	−0.0243	0.5839	0.5875
	60	−0.0312	0.5801	0.5823
	80	−0.0352	0.5771	0.5801
	100	−0.0374	0.5902	0.5929
	120	−0.0363	0.5883	0.5901
4	20	−0.0086	0.6038	0.5942
	40	−0.0281	0.6017	0.6017
	60	−0.0418	0.5909	0.5842
	80	−0.0456	0.5810	0.5783
	100	−0.0497	0.5803	0.5815
	120	−0.0544	0.5708	0.5755
6	20	−0.0064	0.6628	0.6659
	40	−0.0308	0.6185	0.6004
	60	−0.0499	0.5681	0.5698
	80	−0.0484	0.5864	0.5849
	100	−0.0663	0.5740	0.5666
	120	−0.0767	0.5463	0.5484
8	20	−0.0053	0.6594	0.6558
	40	−0.0491	0.5694	0.5703
	60	−0.0502	0.5904	0.5930
	80	−0.0764	0.5326	0.5282
	100	−0.0763	0.5577	0.5647
	120	−0.0978	0.5245	0.5113
10	20	−0.0192	0.6055	0.6072
	40	−0.0388	0.5704	0.5653
	60	−0.0723	0.5109	0.5082
	80	−0.0996	0.4983	0.4919
	100	−0.0996	0.5036	0.4976
	120	−0.1067	0.4945	0.5001
12	20	−0.0198	0.6008	−0.0187
	40	−0.0323	−0.0143	−0.0126
	60	−0.0917	0.4929	−0.0418
	80	−0.0924	0.5278	−0.0503
	100	−0.0905	0.5271	−0.0424
	120	−0.1261	0.4602	−0.0817

For a better understanding of the performance of the algorithms, boxplots showing the makespan (C_{max}) distribution in terms of N for different values of M are depicted in Figures 1–3 for Balanced, Dominant S, and Dominant P instances, respectively. Using this visualization, it is more perceptible that when the problem becomes more complex (that is, the number of tasks increases), the makespan value also increases. It can also be concluded that most of the GA results are better than the ACOII method.

In Table 6, the average execution time, in seconds, for the genetic algorithm when solving Balanced, Dominant P, and Dominant S large problems is presented. It can be seen that as the number of tasks increases for each number of concurrent machines, the execution time also increases. The lowest average execution time value is observed for problems with six machines. Conversely, the highest average execution time value is obtained for problems with four concurrent machines. It is also observed that the highest

average execution time occurred for the Balanced problem with $M = 2$ and $N = 120$. For the Dominant P problem with $M = 2$ and $N = 20$, the shortest average execution time was obtained.

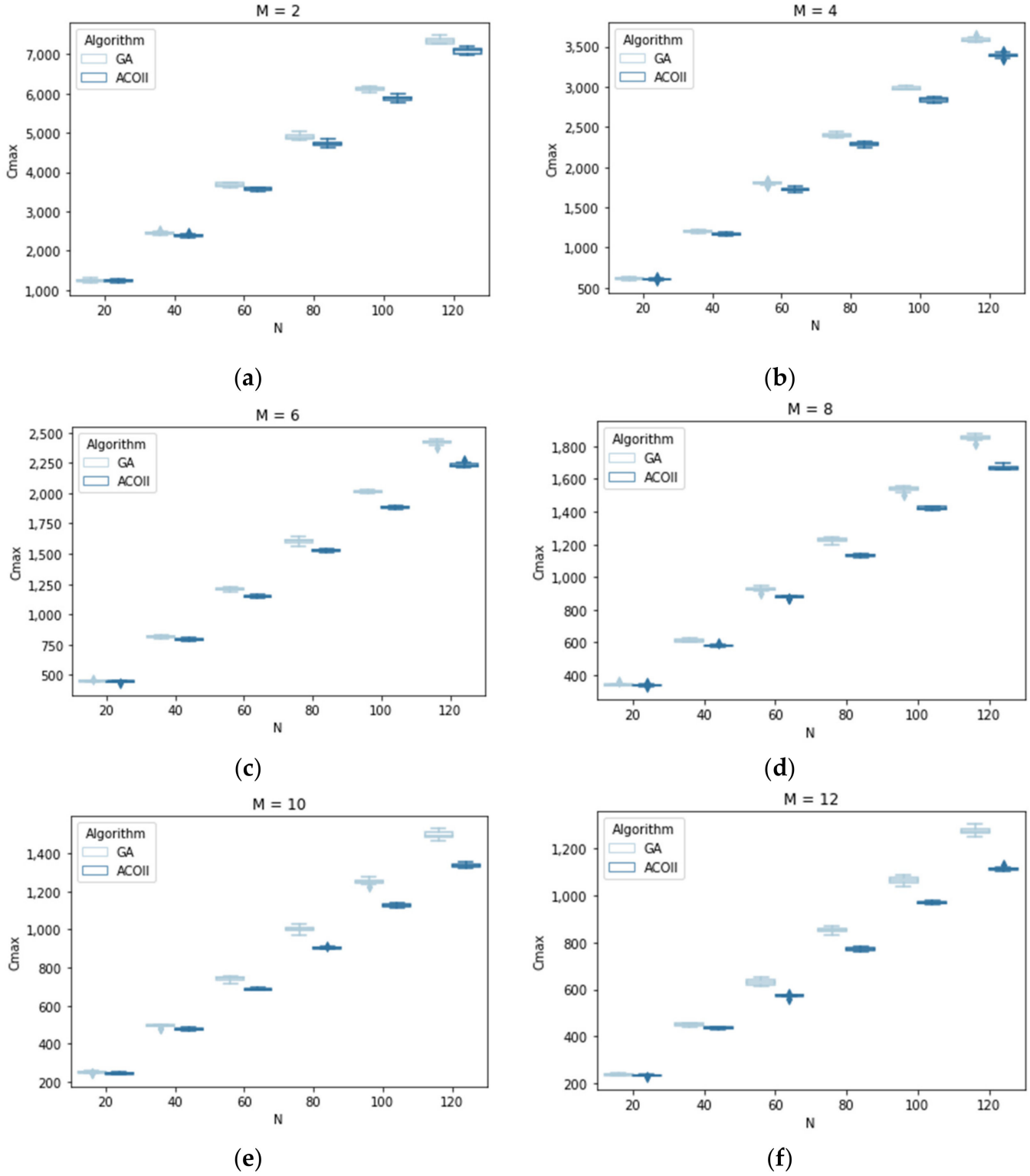

Figure 1. Makespan boxplot for Balanced large problems for (**a**) $M = 2$; (**b**) $M = 4$; (**c**) $M = 6$; (**d**) $M = 8$; (**e**) $M = 10$; (**f**) $M = 12$.

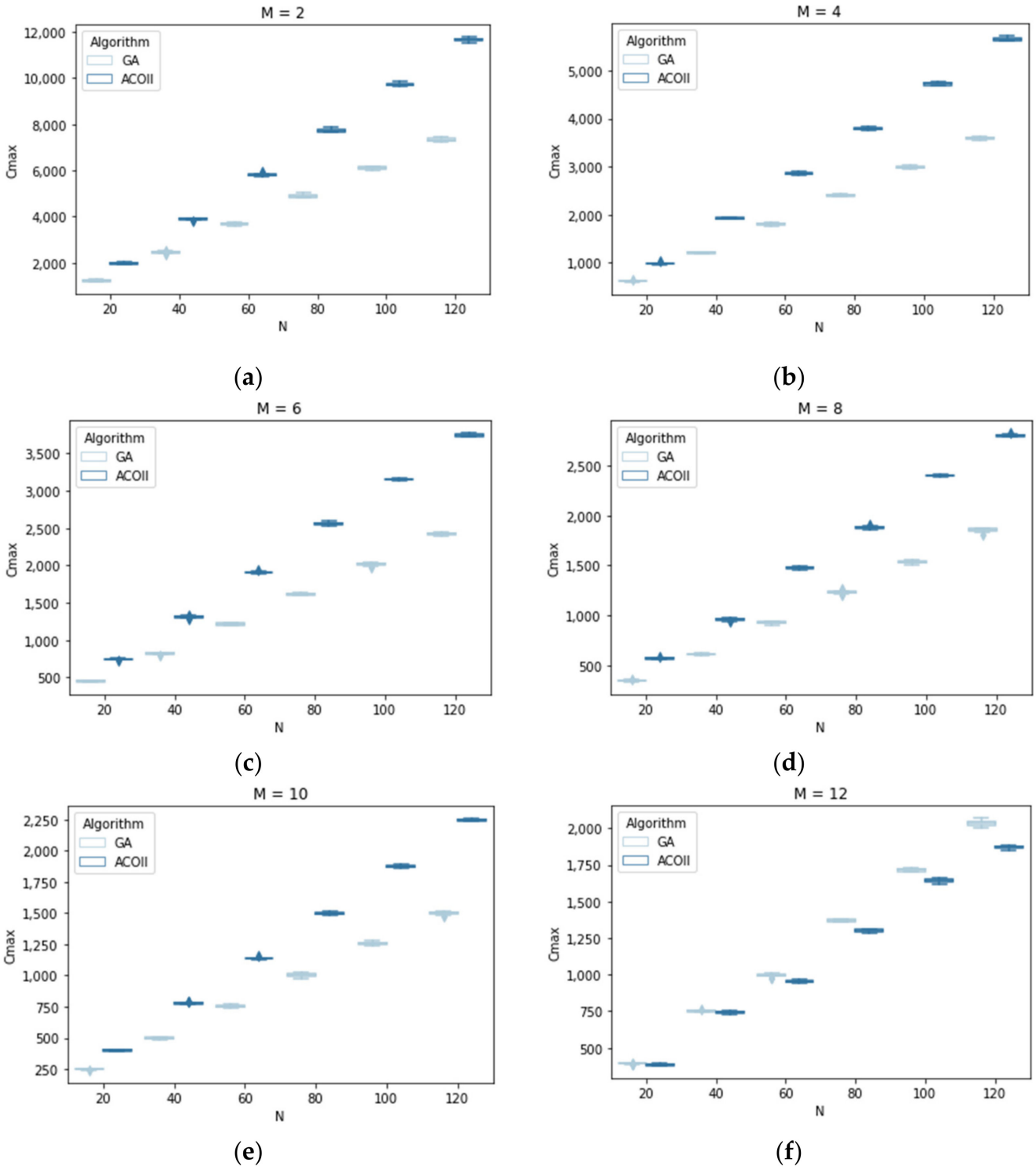

Figure 2. Makespan boxplot for Dominant P large problems for (**a**) $M = 2$; (**b**) $M = 4$; (**c**) $M = 6$; (**d**) $M = 8$; (**e**) $M = 10$; (**f**) $M = 12$.

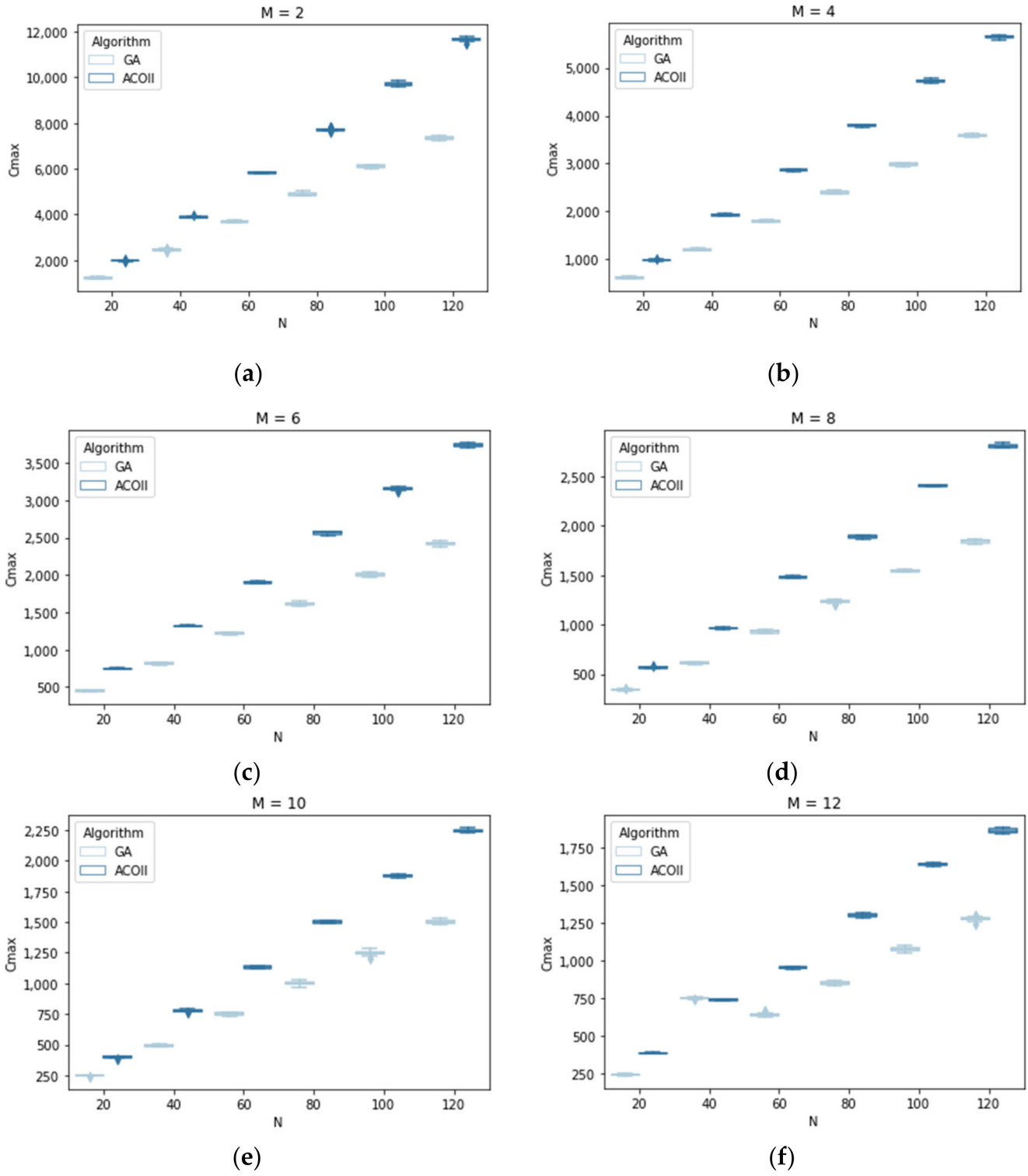

Figure 3. Makespan boxplot for Dominant S large problems for (**a**) $M = 2$; (**b**) $M = 4$; (**c**) $M = 6$; (**d**) $M = 8$; (**e**) $M = 10$; (**f**) $M = 12$.

Table 6. Average GA execution time for Balanced, Dominant P, and Dominant S for large problems.

			Number of Tasks (N)					
			20	**40**	**60**	**80**	**100**	**120**
Number of Machines (M)	Balanced	2	19.95	29.97	37.06	63.74	81.56	110.90
		4	22.24	35.65	50.56	60.74	85.44	94.31
		6	25.11	33.63	45.22	53.70	73.65	86.47
		8	26.74	35.38	41.47	50.82	60.51	71.67
		10	31.36	39.80	47.78	60.42	71.74	83.71
		12	31.20	40.78	50.03	53.02	61.63	69.19
	Dominant P	2	14.74	21.25	31.71	44.08	61.96	84.73
		4	16.93	23.22	30.66	41.56	51.82	65.73
		6	17.85	24.70	30.51	37.91	46.17	59.19
		8	25.12	33.38	42.34	50.66	59.03	70.64
		10	30.72	38.78	46.42	54.00	63.82	73.47
		12	32.47	40.64	49.22	57.69	65.36	73.28
	Dominant S	2	14.84	20.96	32.30	45.63	60.52	84.44
		4	17.17	23.37	32.06	42.02	49.55	64.90
		6	18.56	24.53	30.68	36.99	45.52	54.41
		8	24.99	33.62	41.27	49.82	57.57	67.14
		10	28.80	39.37	48.03	52.45	59.35	69.27
		12	31.11	36.81	44.94	51.85	59.18	64.69

6. Conclusions

Decision support systems are a much-needed factor for industries worldwide. For this, raw material production scheduling and mathematical algorithms are widely used methods to help companies make the best decisions. These decisions avoid unnecessary order cancellations or delays and help to better manage product production costs. In parallel machine scheduling problems, there is a need to assign tasks to machines, along with sequencing problems. In this article, a scheduling problem was developed in an environment of unrelated parallel machines. The machines are considered unrelated when the processing times of tasks depend on the machines to which they are assigned and when there is no relationship between the speeds of the machines. Two types of problems were applied, small and large problems (Balanced, Dominant S, and Dominant P), using the genetic algorithm to minimize the makespan.

In terms of small problems, in general, the proposed approach obtained good results in terms of the makespan, achieving the known optimal value. For Dominant S ($M = 4, N = 7$) and Dominant P ($M = 4, N = 6$) problems, the optimal solution was not found, despite the makespan of the solutions found being very close to the optimal value. Moreover, a new comparison was made with the ACOII method, and there are nine values where the genetic algorithm achieved better results and only four values where ACOII performed better. However, after applying hypothesis testing, there were no significant differences between GA and ACOII results, considering a level of confidence of 95%. The average execution time was the longest for the Dominant S problems when $M = 8$ and $N = 11$. However, the shortest average execution time was obtained for Dominant P problems for $M = 2$ and $N = 6$.

For large problems, the optimal solution is unknown, and thus, a comparison was made between the solutions obtained by the genetic algorithm and the ACOII method. For the Balanced problems, the genetic algorithm exhibited the worst performance. On the contrary, for both Dominant S and Dominant P problems, the genetic algorithm performed better, except when $M = 12$ and $N = 40$ and for $M = 12$, respectively. Furthermore, it was proved that there are significant differences between the average makespan values of the genetic algorithm and the ACOII methods. Moreover, it was also possible to observe that the highest average execution time occurred for the Balanced problem with $M = 2$

and $N = 120$. For the Dominant P problem with $M = 2$ and $N = 20$, the shortest average execution time was obtained.

In conclusion, GA has the ability to effectively solve small and large scheduling problems when minimizing the makespan of unrelated parallel machines with sequence-dependent setup times; better performance for the GA was observed in the comparative statistical analysis between the two metaheuristics, especially for the large problems.

In the future, we intend to study the effect of using different chromosome representations and genetic operators in GA performance and to consider other objectives such as completion time and tardiness.

Author Contributions: Conceptualization, L.R.V., A.M.A.C.R. and L.A.C.; methodology, A.M.A.C.R., L.R.V., L.A.C., A.R.A. and M.A.M.; software, A.M.A.C.R., L.A.C., A.R.A. and M.A.M.; validation, A.R.A., A.M.A.C.R., L.A.C., L.R.V. and M.A.M.; formal analysis, A.R.A., A.M.A.C.R., M.A.M. and L.A.C.; investigation, L.R.V., A.R.A., A.M.A.C.R., L.A.C. and M.A.M.; resources, A.M.A.C.R., L.A.C. and L.R.V.; data curation, A.R.A., A.M.A.C.R., L.A.C., L.R.V. and M.A.M.; writing—original draft preparation, A.R.A., L.R.V. and M.A.M.; writing—review and editing, A.R.A., A.M.A.C.R., L.A.C., L.R.V. and M.A.M.; visualization, A.R.A., A.M.A.C.R., L.A.C., L.R.V. and M.A.M.; supervision, A.M.A.C.R., L.A.C. and L.R.V.; project administration, A.M.A.C.R., L.A.C. and L.R.V.; funding acquisition, A.M.A.C.R., L.A.C. and L.R.V. All authors have read and agreed to the published version of the manuscript.

Funding: The project is funded by the FCT—Fundação para a Ciência e Tecnologia through the R&D Units Project Scope UIDB/00319/2020 and EXPL/EME-SIS/1224/2021 and PhD grant UI/BD/150936/2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Schuh, G.; Reuter, C.; Prote, J.-P.; Brambring, F.; Ays, J. Increasing data integrity for improving decision making in production planning and control. *CIRP Ann.* **2017**, *66*, 425–428. [CrossRef]
2. Pinedo, M.L. *Scheduling Theory, Algorithms, and Systems*, 5th ed.; Springer: New York, NY, USA, 2016.
3. Santos, A.S.; Madureira, A.M.; Varela, M.L.R. An ordered heuristic for the allocation of resources in unrelated paral-lel-machines. *Int. J. Ind. Eng. Comput.* **2015**, *6*, 145–156.
4. Su, L.-H.; Cheng, T.; Chou, F.-D. A minimum-cost network flow approach to preemptive parallel-machine scheduling. *Comput. Ind. Eng.* **2013**, *64*, 453–458. [CrossRef]
5. Tan, Z.; Chen, Y.; Zhang, A. Parallel machines scheduling with machine maintenance for minsum criteria. *Eur. J. Oper. Res.* **2011**, *212*, 287–292. [CrossRef]
6. Vallada, E.; Ruiz, R. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *Eur. J. Oper. Res.* **2011**, *211*, 612–622. [CrossRef]
7. Rabadi, G.; Moraga, R.J.; Al-Salem, A. Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times. *J. Intell. Manuf.* **2006**, *17*, 85–97. [CrossRef]
8. 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.* **2009**, *21*, 693–701. [CrossRef]
9. 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.* **2012**, *25*, 43–53. [CrossRef]
10. Yang, Q.; Guo, X.; Gao, X.-D.; Xu, D.-D.; Lu, Z.-Y. Differential Elite Learning Particle Swarm Optimization for Global Numerical Optimization. *Mathematics* **2022**, *10*, 1261. [CrossRef]
11. Leung, M.-F.; Coello, C.A.C.; Cheung, C.-C.; Ng, S.-C.; Lui, A.K.-F. A Hybrid Leader Selection Strategy for Many-Objective Particle Swarm Optimization. *IEEE Access* **2020**, *8*, 189527–189545. [CrossRef]
12. Das, S.; Suganthan, P.N. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
13. Pan, Q.-K.; Tasgetiren, M.F.; Liang, Y.-C. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Comput. Ind. Eng.* **2008**, *55*, 795–816. [CrossRef]

14. Ho, M.H.; Hnaien, F.; Dugardinr, F. Exact method to optimize the total electricity cost in two-machine permutation flow shop scheduling problem under Time-of-use tariff. *Comput. Oper. Res.* **2022**, *144*, 105788. [CrossRef]
15. Foumani, M.; Razeghi, A.; Smith-Miles, K. Stochastic optimization of two-machine flow shop robotic cells with con-trollable inspection times: From theory toward practice. *Robot. Comput.-Integr. Manuf.* **2020**, *61*, 101822. [CrossRef]
16. Artiba, A.; Elmaghraby, S.E. *The Planning and Scheduling of Production Systems*; Springer Science & Business Media: Berlin, Germany, 1996. [CrossRef]
17. Brucker, P. Due-date scheduling. In *Scheduling Algorithms*; Springer: Berlin/Heidelberg, Germany, 2001. [CrossRef]
18. McNaughton, R. Scheduling with Deadlines and Loss Functions. *Manag. Sci.* **1959**, *6*, 1–12. [CrossRef]
19. Bülbül, K.; Şen, H. An exact extended formulation for the unrelated parallel machine total weighted completion time problem. *J. Sched.* **2016**, *20*, 373–389. [CrossRef]
20. Nikabadi, M.S.; Naderi, R. A hybrid algorithm for unrelated parallel machines scheduling. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 681–702. [CrossRef]
21. Reddy, M.S.; Ratnam, C.; Agrawal, R.; Varela, M.L.; Sharma, I.; Manupati, V. Investigation of reconfiguration effect on makespan with social network method for flexible job shop scheduling problem. *Comput. Ind. Eng.* **2017**, *110*, 231–241. [CrossRef]
22. Varela, M.L.R.; Silva, S.D.C. An ontology for a model of manufacturing scheduling problems to be solved on the web. In Proceedings of the International Conference on Information Technology for Balanced Automation Systems, Porto, Portugal, 23–25 June 2008; Springer: Boston, MA, USA, 2008; pp. 197–204.
23. Woo, Y.-B.; Jung, S.; Kim, B.S. A rule-based genetic algorithm with an improvement heuristic for unrelated parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Comput. Ind. Eng.* **2017**, *109*, 179–190. [CrossRef]
24. Xu, L.; Wang, Q.; Huang, S. Dynamic order acceptance and scheduling problem with sequence-dependent setup time. *Int. J. Prod. Res.* **2015**, *53*, 1–12. [CrossRef]
25. Zhang, S.; Wong, T. Studying the impact of sequence-dependent set-up times in integrated process planning and scheduling with E-ACO heuristic. *Int. J. Prod. Res.* **2015**, *54*, 4815–4838. [CrossRef]
26. Allahverdi, A. The third comprehensive survey on scheduling problems with setup times/costs. *Eur. J. Oper. Res.* **2015**, *246*, 345–378. [CrossRef]
27. Baker, K.R.; Trietsch, D. *Principles of Sequencing and Scheduling*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2009. [CrossRef]
28. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Kan, A.H.G.R. *Optimization and Approximation in Deterministic Se-Quencing and Scheduling*; Elsevier: Amsterdam, The Netherlands, 1979; Volume 5, pp. 287–326.
29. Blazewicz, J.; Domschke, W.; Pesch, E. The job shop scheduling problem: Conventional and new solution techniques. *Eur. J. Oper. Res.* **1996**, *93*, 1–33. [CrossRef]
30. Błażewicz, J.; Machowiak, M.; Węglarz, J.; Kovalyov, M.Y.; Trystram, D. Scheduling Malleable Tasks on Parallel Processors to Minimize the Makespan. *Ann. Oper. Res.* **2004**, *129*, 65–80. [CrossRef]
31. Lin, S.-W.; Lu, C.-C.; Ying, K.-C. Minimization of total tardiness on unrelated parallel machines with sequence- and machine-dependent setup times under due date constraints. *Int. J. Adv. Manuf. Technol.* **2010**, *53*, 353–361. [CrossRef]
32. Zeidi, J.R.; MohammadHosseini, S. Scheduling unrelated parallel machines with sequence-dependent setup times. *Int. J. Adv. Manuf. Technol.* **2015**, *81*, 1487–1496. [CrossRef]
33. Jungwattanakit, J.; Reodecha, M.; Chaovalitwongse, P.; Werner, F. A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Comput. Oper. Res.* **2009**, *36*, 358–378. [CrossRef]
34. Gendreau, M.; Laporte, G.; Guimarães, E.M. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *Eur. J. Oper. Res.* **2001**, *133*, 183–189. [CrossRef]
35. Kim, D.-W.; Kim, K.-H.; Jang, W.; Chen, F.F. Unrelated parallel machine scheduling with setup times using simulated annealing. *Robot. Comput. Manuf.* **2002**, *18*, 223–231. [CrossRef]
36. Tang, L.; Wang, X. Simultaneously scheduling multiple turns for steel color-coating production. *Eur. J. Oper. Res.* **2009**, *198*, 715–725. [CrossRef]
37. Pfund, M.; Fowler, J.W.; Gupta, J.N.D. A Survey Of Algorithms For Single And Multi-Objective Unrelated Parallel-Machine Deterministic Scheduling Problems. *J. Chin. Inst. Ind. Eng.* **2004**, *21*, 230–241. [CrossRef]
38. Kim, D.-W.; Na, D.-G.; Chen, F.F. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robot. Comput. Manuf.* **2003**, *19*, 173–181. [CrossRef]
39. Ghirardi, M.; Potts, C. Makespan minimization for scheduling unrelated parallel machines: A recovering beam search approach. *Eur. J. Oper. Res.* **2005**, *165*, 457–467. [CrossRef]
40. Zheng, X.-L.; Wang, L. A Collaborative Multiobjective Fruit Fly Optimization Algorithm for the Resource Constrained Unrelated Parallel Machine Green Scheduling Problem. *IEEE Trans. Syst. Man, Cybern. Syst.* **2016**, *48*, 790–800. [CrossRef]
41. Aydilek, A.; Aydilek, H.; Allahverdi, A. Minimising maximum tardiness in assembly flowshops with setup times. *Int. J. Prod. Res.* **2017**, *55*, 7541–7565. [CrossRef]
42. Abreu, L.; Prata, B. A Hybrid Genetic Algorithm for Solving the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times. *IEEE Lat. Am. Trans.* **2018**, *16*, 1715–1722. [CrossRef]
43. Gedik, R.; Kalathia, D.; Egilmez, G.; Kirac, E. A constraint programming approach for solving unrelated parallel machine scheduling problem. *Comput. Ind. Eng.* **2018**, *121*, 139–149. [CrossRef]

44. Fanjul-Peyro, L.; Perea, F.; Ruiz, R. Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *Eur. J. Oper. Res.* **2017**, *260*, 482–493. [CrossRef]
45. Arnaout, J.-P. A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Ann. Oper. Res.* **2019**, *285*, 273–293. [CrossRef]
46. Amaral, G.; Costa, L.; Rocha, A.M.A.C.; Varela, L.; Madureira, A. Application of the Simulated Annealing Algorithm to Minimize the makespan on the Unrelated Parallel Machine Scheduling Problem with Setup Times. In *International Conference on Hybrid Intelligent Systems*; Springer: Cham, Switzerland, 2019; pp. 398–407. [CrossRef]
47. Rabadi, G. (Ed.) *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling*; Springer: Berlin, Germany, 2016; Volume 236.
48. Guinet, A. Textile production systems: A succession of non-identical parallel processor shops. *J. Oper. Res. Soc.* **1991**, *42*, 655–671. [CrossRef]
49. Scheduling Research Virtual Center Homepage. Available online: www.SchedulingResearch.com (accessed on 17 January 2022).
50. Holland, J.H. *Adaptation in Natural and Artificial Systems*, 1st ed.; University of Michigan Press: Ann Arbor, MI, USA, 1975.
51. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
52. Lambora, A.; Gupta, K.; Chopra, K. Genetic Algorithm—A Literature Review. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 380–384.
53. Teoh, T.T.; Rong, Z. Python for Data Analysis. In *Artificial Intelligence with Python*; Springer: Singapore, 2022; pp. 107–122. [CrossRef]
54. Bonald, T.; de Lara, N.; Lutz, Q.; Charpentier, B. Scikit-network: Graph analysis in python. *J. Mach. Learn Res.* **2020**, *21*, 1–6.
55. Montgomery, D.C.; Runger, G.C. *Applied Statistics and Probability for Engineers*; Wiley: Hoboken, NJ, USA, 2018; p. 710.
56. Waskom, M.L. Seaborn: Statistical data visualization. *J. Open Source Softw.* **2021**, *6*, 3021. [CrossRef]