sustainability

MDPI

Article

Green Scheduling of Identical Parallel Machines with Release Date, Delivery Time and No-Idle Machine Constraints

Lotfi Hidri [1,*], Ali Alqahtani [1], Achraf Gazdar [2] and Belgacem Ben Youssef [3]

[1] Industrial Engineering Department, College of Engineering, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia; 439106478@student.ksu.edu.sa
[2] Software Engineering Department, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia; agazdar@ksu.edu.sa
[3] Computer Engineering Department, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia; bbenyoussef@ksu.edu.sa
* Correspondence: lhidri@ksu.edu.sa

Abstract: Global warming and climate change are threatening life on earth. These changes are due to human activities resulting in the emission of greenhouse gases. This is caused by intensive industrial activities and excessive fuel energy consumption. Recently, the scheduling of production systems has been judged to be an effective way to reduce energy consumption. This is the field of green scheduling, which aims to allocate jobs to machines in order to minimize total costs, with a focus on the sustainable use of energy. Several studies have investigated parallel-machine shops, with a special focus on reducing and minimizing the total consumed energy. Few studies explicitly include the idle energy of parallel machines, which is the energy consumed when the machines are idle. In addition, very few studies have considered the elimination of idle machine times as an efficient way to reduce the total consumed energy. This is the no-idle machine constraint, which is the green aspect of the research. In this context, this paper addresses the green parallel-machine scheduling problem, including release dates, delivery times, and no-idle machines, with the objective of minimizing the maximum completion time. This problem is of practical interest since it is encountered in several industry processes, such as the steel and automobile industries. A mixed-integer linear programming (MILP) model is proposed for use in obtaining exact solutions for small-sized instances. Due to the NP-hardness of the studied problem, and encouraged by the successful adaptation of metaheuristics for green scheduling problems, three genetic algorithms (GAs) using three different crossover operators and a simulated annealing algorithm (SA) were developed for large-sized problems. A new family of lower bounds is proposed. This was intended for the evaluation of the performance of the proposed algorithms over the average percent of relative deviation (ARPD). In addition, the green aspect was evaluated over the percentage of saved energy, while eliminating the idle-machine times. An extensive experimental study was carried out on a benchmark of test problems with up to 200 jobs and eight machines. This experimental study showed that one GA variant dominated the other proposed procedures. Furthermore, the obtained numerical results provide strong evidence that the proposed GA variant outperformed the existing procedures from the literature. The experimental study also showed that the adoption of the no-idle machine time constraints made it possible to reduce the total consumed energy by 29.57%, while the makespan (cost) increased by only 0.12%.

Keywords: green scheduling; parallel machines; release time; delivery time; no-idle time constraint; metaheuristic; mixed-integer linear program

check for updates

Citation: Hidri, L.; Alqahtani, A.; Gazdar, A.; Ben Youssef, B. Green Scheduling of Identical Parallel Machines with Release Date, Delivery Time and No-Idle Machine Constraints. *Sustainability* **2021**, *13*, 9277. https://doi.org/10.3390/su13169277

Academic Editors: Lin Li, Karl R. Haapala, Sophie I. Hallstedt and Yiran (Emma) Yang

Received: 25 April 2021
Accepted: 5 August 2021
Published: 18 August 2021

1. Introduction

In recent decades, impressive climate change has been observed. This is due to greenhouse gas emission. These greenhouse gas emissions are strongly related to excessive industrial activities and fuel energy consumption. According to the US Energy Information

Administration, the consumed energy in the industrial sector is approximately 54% of the total consumed energy [1]. Consequently, production planners should take into account these environmental issues when preparing their plans. The common way to struggle against these environmental concerns is the efficient use of energy in manufacturing industries [2]. Several recent studies have presented scheduling approaches as important tools to improve energy efficiency and to reduce consumed energy in manufacturing industries [3,4].

It is noteworthy that scheduling is the allocation of resources (processors or machines) to jobs (or tasks) within a given time window in order to optimize one or several objectives [5]. The scheduling strategies intended to reduce energy consumption and/or improve energy efficiency are called "energy-efficient scheduling" or "green scheduling problems" (GSPs) [2]. Therefore, GSPs involve the allocation of jobs to machines in order to minimize the total cost (makespan, tardiness, etc.) and consumed energy and/or improve the energy efficiency [1]. Thus, compared to classical scheduling, energy-efficient scheduling takes into account the environmental issues over saving the consumed energy.

In GSPs, there are two ways to consider the minimization of consumed energy: (1) by explicitly including the energy in the objective function [2], or (2) by including the energy consumption as a constraint [6–8]. In this study, the second method is adopted through a special focus on the machine idle times. Idle machine times are periods when a machine is ready for processing jobs but there is no job to be treated. During these idle periods, a machine consumes energy without providing any services. In addition, it has been observed that machines in service stay in idle mode most of the time and, during this idle phase, 80% of the total energy is consumed [9]. This energy is referred to as idle energy. Therefore, energy could be saved by better controlling the idle machine periods [10].

In order to save idle energy, two approaches can be adopted. The first one consists of shutting down the idle machines. This method is applicable for electric machines where moving from an idle state to an active state is easy [11]. For other shop environments, such as for furnaces [12], the first strategy is not applicable. Indeed, for such shops, turning off the machines and then turning them on again is highly energy consuming. Alternatively, no-idle machine time constraints can be adopted. These no-idle machine constraints consist of forcing each machine to process all the assigned jobs continuously, in order to avoid idle times. In this study, no-idle machine constraints were adopted: this is the energy-saving solution that comprises the green aspect of the treated problem.

GSPs are important from a practical point of view because they model several real-life and practical production processes pertaining to environmental issues. In this context, real-life scheduling problems have been addressed for furnaces used in the steel [12–14] and automobile industries [15]. These production processes are intended to produce different steel products. The shop is composed of identical parallel furnaces to avoid a single-furnace bottleneck. In addition, the pieces requiring processing in the furnaces arrive at the shop on different dates; these are the release dates. After finishing processing, a produced piece is still hot and requires a certain time for cooling; this is the delivery time. The objective is to minimize the maximum completion time (cost) and reduce the furnaces' high consumption of energy by eliminating the idle machine times.

Motivated by the abovementioned real-life problems and their practical relevance in the steel and car industries, this study addresses a green scheduling problem. This problem is the parallel machine scheduling problem with release dates, delivery times, and no-idle machines. With no-idle machines, no idle time is permitted between two consecutive processed jobs in any machine. By preventing machines from having idle times, the total consumed energy is reduced, and the green requirement is satisfied. This is the green aspect for this scheduling problem. The objective of this is to determine a feasible schedule without idle machine times that minimizes the maximum completion time (makespan). This problem occurs frequently in plants using furnaces with high energy consumption [14]. The objective of this study was to develop methods that allow optimal or near optimal solutions to be obtained for the studied green scheduling problem.

GSPs are a subclass of production scheduling problems, which are proven to be NP-hard in the majority of cases; therefore, this hard complexity also holds for the GSPs. The procedures solving the hard scheduling problems consist of three categories: the exact methods, the heuristics, and the metaheuristics (evolutionary: genetic algorithm (GA) and simulated annealing (SA) [5]. The heuristics and metaheuristics algorithms are called approximate methods. The two main criteria for selecting the appropriate method to be used are the computational time and the solution quality. The exact methods are known to be time consuming whilst they solve the problems. Due to this drawback, these methods are generally used to solve small-sized problems. The heuristics methods are characterized by a low computational time and a moderate solution quality. The metaheuristics are reputed to have an acceptable computational time and a near optimal solution [5,16,17]. According to a recent survey study [2], almost 59% of published research works for GSP successfully utilize the evolutionary algorithms (metaheuristics) as a solving approach. Encouraged by these successes, this study investigates the usage of metaheuristics, such as the GA and SA, in solving the problem in question.

The main contributions of this work are presented as follows: (1) The studying of the green parallel machine scheduling problem with release dates, delivery times, and no-idle machines; (2) the development of new efficient lower bounds on the makespan for the studied problem; (3) the successful adaptation of some metaheuristics, GA and SA, to efficiently solve the problem in question. These methods outperform the existing procedures in solving the problem, as shown in the experimental study section.

The remainder of this paper is organized as follows. In Section 2, an exhaustive literature review is presented. In Section 3, the studied problem is defined, some of its useful proprieties are proposed, and new lower bounds are developed. A mixed integer linear formulation is proposed in Section 4. The proposed metaheuristics are the topic of Section 5. An extensive experimental study, intended to assess the performance of the proposed procedures, is presented in Section 6. Finally, a conclusion summarizing the main findings and presenting future research directions is presented.

2. Literature Review

In this section, the classical parallel machine scheduling problem and some of its important variants are briefly reviewed. Afterwards, the GSP-related literature focusing on parallel machines is presented. This literature mainly discusses the consideration of energy consumption, with a focus on idle energy consumption. Finally, the research gap is deduced.

2.1. Classic Parallel Machine Scheduling Problem

In the classical deterministic identical parallel machine problem, there are a number of independent jobs to be processed in a range of identical machines. Each job has to be carried out on one of the machines during a fixed processing time, without preemption. Finding the schedule that minimizes the maximum completion time (makespan) is considered as the objective for this problem. For more than 50 years, the study of identical parallel scheduling has been considered to be one of the most important types of scheduling. Many parallel scheduling problems have been proved to be NP-hard [18]. Polynomial algorithms are not, therefore, likely to be used to achieve optimal solutions; thus, several heuristics and metaheuristics were investigated, such as in [19–21]. To determine near optimal solution for these problems, many polynomial-time algorithms have been suggested. The longest processing time (LPT) rule was proposed in [22] and has received significant attention. It solves single-criterion makespan problems and is one of the earliest scheduling guidelines for parallel machine scheduling problems. It is a priority rule, and the job is placed in a non-increasing processing time order, and the next task on the list is scheduled.

In the literature, loading in the identical parallel machine was linked to different programming criteria, even if the workload imbalance was considered a restriction or a purpose. The authors in [23] proposed a dynamic neural network that uses time to solve the scheduling problem with different penalty parameters. The simulation tests have shown that all data sets are covered by the proposed network and are less effective than LPT. The authors in [24] proposed a neural net algorithm (NNA) to minimize makespan on parallel processing machines with identical job sizes with the introduction of a master weight matrix (MWM) and new methods of coding. The result is remarkably effective compared with other heuristics, especially in large-scale instances. In [25], the authors proposed a genetic algorithm (GA) based on a heuristic procedure with SPT and LPT rules to minimize the workload unbalance between the machines and reduce the makespan. Due to the accuracy and speed of the solution, the authors found better results than those for the strategies proposed in their first study [26].

The authors in [27] presented an SA approach solving the same parallel machinery make-up problem. The findings show that the SA algorithm is very successful even for large problems. This is because the execution time is less than one second for all generated test problems. Thus, in the form of complicated scheduling problems, the SA approach is worth considering. A mixed-integer linear programming method was developed by [28] and solved using CPLEX for small to moderately large-scale problems, with various availability limitations on all machines. The authors used lexicographic enumeration algorithms to solve major problems. Dominance rules are proposed for reducing the search space and increasing the efficiency of the algorithm. To formulate an identical parallel machine scheduling problem, in [29], the authors used an integer linear programming model (ILP), which aims to reduce the total relative imbalance, and the most extended processing time algorithm is used to find an initial solution. The findings showed that mathematical modeling and algorithms are essential tools and are more efficient than conventional approaches for these problems.

2.2. Green Parallel Machine Scheduling Problem with Energy Consumption Consideration

Green parallel machine scheduling problems with energy consumption consideration were addressed in the literature, and in the following, the most recent papers are presented. In [30], a green parallel machine scheduling problem was studied, with the simultaneous minimization of energy consumption and makespan. Greedy heuristic and local search procedures were developed to solve the latter problem. Experimental results show the efficiency of the proposed procedures. The green uniform parallel machine scheduling problem was examined in [31], where the consumed energy and makespan were optimized simultaneously as a bi-objective function. Several efficient heuristics were proposed and tested on a benchmark of instances.

An unrelated parallel machine scheduling problem with an energy efficiency consideration was addressed in [32]. A mixed integer linear program (MILP), dominance rules, and a heuristic algorithm were developed to solve the latter problem. In [33], the unrelated parallel machine scheduling problem, with the optimization of total tardiness and energy consumption, was addressed. GA, Cat Swarm Optimization and Interactive Artificial Bee Colony were proposed to solve the studied problem. A green identical parallel machine scheduling problem, with simultaneous minimization of the total consumed energy and makespan, was studied in [15]. A local strategy search was applied to obtain satisfactory solutions.

The authors in [34] studied an identical parallel machine scheduling problem with energy consumption consideration. The energy consumption is introduced as a constraint. Efficient heuristics were proposed to solve this problem. In [35], the parallel dedicated machine problem with energy consumption limits was addressed. First, complexity results were provided. Mixed-integer linear programming and a local search heuristic were developed to solve the studied problem. The experimental study shows the efficiency of the proposed heuristic. An identical parallel machine problem with simultaneous minimization of the consumed energy and the makespan was addressed in [15]. In the latter study, a local search-based constructive heuristic was proposed. The effectiveness of the presented heuristic was proofed over an extensive experimental study. The authors in [36] studied a parallel identical batch-processing machine scheduling problem with dynamic arrival jobs. The makespan and electric consumed energy are simultaneously minimized through a bi-objective function. An ant colony-based algorithm was proposed to identify the optimal Pareto. In addition, a local optimization-based heuristic was presented. The experimental study provided proof of the efficiency of the proposed procedures.

An unrelated parallel machine scheduling problem, with the maximum energy consumption constraint, was considered in [37]. A three-stage heuristic was proposed to provide a near optimal solution. In the first and second stages, the consumed energy was minimized. The third stage focused on makespan minimization. The experimental study shows that the developed methods outperform the existing procedures.

The authors in [32] were interested in an unrelated parallel machine scheduling problem with energy consumption consideration. The objective is the energy cost minimization. A mixed integer linear program was proposed. Several inequalities and dominance rules were presented to enhance the performance of the proposed program. In addition, a heuristic algorithm was proposed for large-sized instances. The experimental study shows that the proposed methods outperform the existing ones.

An unrelated parallel machine problem, with simultaneous minimization of the consumed energy and makespan, was considered in [38]. In order to provide a near optimal solution, a mimetic differential evolution algorithm was proposed. This algorithm was enhanced by embedding a local search-based heuristic. Experimental results show that the proposed algorithm outperforms the existing procedures.

The authors in [39] studied a parallel machine scheduling problem with an energy consumption budget. In other terms, the energy consumption is considered as a constraint. The net revenue is the objective function to be maximized. A modified variable neighborhood search algorithm was presented. The computational results show that the presented algorithm outperforms existing methods.

An unrelated parallel machine scheduling problem was considered in [33]. The objective function to be minimized is a linear combination of the consumed energy and the mean weighted tardiness. Three metaheuristics were presented to solve the studied problem. The obtained solutions were compared with exact solutions for small-sized instances, and the results are satisfactory.

In [40], a parallel machine scheduling problem, with a reduction in the total electricity demand, was addressed. A mathematical optimization program was proposed to obtain schedules minimizing the total consumed energy. Computational results show the efficiency of the proposed algorithm.

A scheduling parallel batch processing machines problem with minimization of makespan and consumed energy was studied in [41]. In order to solve the latter problem, a metaheuristic with new features was proposed. The intensive experimental study shows that the presented metaheuristic outperforms the existing algorithms.

2.3. Green Parallel Machine Scheduling Problem with Idle Energy Consideration

In the previous references, the consumed energy in GSP is totally taken into account either in the objective function or as a constraint. In other references, the focus is placed on idle machine energy. This kind of energy is consumed while the machine is idle. Eliminating or reducing this kind of energy is seen as an effective way to reduce the total consumed energy [2]. In the following, a literature review focused on idle machine energy is presented.

In [42], an unrelated parallel machine problem with energy consumption consideration was addressed. The consumed energy and the total tardiness are minimized simultaneously. The total consumed energy is explicitly expressed as the sum of setup energy and idle energy. A mathematical program and an enhanced ant colony algorithm were proposed. The experimental study shows that the proposed methods are efficient and outperform the existing ones.

The authors in [7] studied an unrelated parallel machine scheduling problem with energy consumption consideration. The idle energy is incorporated explicitly within the total consumed energy. An iterated local search algorithm was proposed to solve the studied problem. The latter problem was encountered in a textile plant.

The research work in [13] was inspired by a steel industry real-life case. The obtained scheduling problem is a parallel machine problem considering release date, deadlines, and energy consumption. In this study, the idle energy was considered explicitly in order to reduce the consumed energy. A mathematical model was presented in order to tackle the latter problem, and the numerical results show that the proposed method outperforms the existing procedure.

The authors in [43] addressed an identical parallel machine scheduling problem considering release dates, deadlines, and energy consumption. The objective is the minimization of the total consumed energy. A machine can switch from a sleep mode to an active mode. In the sleep mode, there is no energy consumption. Moving from the sleep mode to the active mode requires energy consumption. An efficient heuristic based on a linear program relaxation was presented.

2.4. Green Parallel Machine Scheduling Problem with Idle Machine Times Elimination

Reducing the consumed energy can be performed by eliminating the idle machine times [2]. In this context, the no-idling constraint is taken into account. The authors in [44] studied the identical parallel machine scheduling problem with unit processing times, release dates, deadlines, and no-idle machines constraints. In the latter study, several theoretical proprieties and an efficient heuristic were presented. Some particular cases for release dates and/or deadlines were addressed, and polynomial algorithms were provided. Lower and upper bounds were developed in addition to an integer linear program. The experimental study shows the efficiency of the proposed methods.

In [45], the parallel machine scheduling problem with release dates, delivery times, and no-idle machine constraints was addressed. Several proprieties of the studied problem were established, and a family of new lower bounds was proposed. A family of efficient heuristics were presented, where some of them are based on an exact procedure, designated for the parallel machine scheduling problem with release dates and delivery times. The intensive experimental study shows the efficiency of the presented procedures.

The parallel machine scheduling problem with release dates, delivery times, and no-idle machine constraints was examined in [46]. In the latter study, authors proposed several dominance rules and heuristics to solve the studied problem. These heuristics are extensions of well-known rules developed originally for the parallel machine scheduling problem with only release date and delivery times. Among these rules are Jackson’s rule and Pott’s algorithm.

2.5. Summary of Literature Review

From the literature review, one can observe that the literature presented for the green parallel machine scheduling problem with energy consumption consideration is satisfactory. However, few research works considered idle energy reduction. In addition, there are very few papers dealing with the no-idle machine times constraint as a means of energy saving, for parallel machine scheduling problems. This is the main motivation behind the study of the identical parallel machine scheduling problem with release dates, delivery times, and no-idle machine times.

3. Problem Definition and Proprieties

In this section, the studied problem is defined, some of its useful proprieties are presented, and a set of lower bounds is proposed.

3.1. Problem Definition

The deterministic parallel machine scheduling problem with release dates, delivery times, and no-idle machines is stated as follows. A set $J = \{J_1, J_2, \ldots, J_n\}$ of n jobs (components) has to be processed on m identical parallel machines (furnaces), with $n > m$. The idle time between two consecutive processed jobs in a machine is not permitted. Indeed, during an idle machine time, a machine is consuming energy without performing any task; this is the idle energy. In a recent study, the idle energy derived from the idle machine times represents 80% of the total consumed energy [9]. Therefore, eliminating the idle machine times is seen as an efficient means to totally eliminate idle energy and consequently to reduce the total consumed energy.

Each job, J_i, has a release date (head or arrival time), r_i, from which the job, J_i, is available for processing in any available machine. Each job, J_i $(1 \leq i \leq n)$, is assigned to a machine where it is processed during p_i. After being processed, a job J_i $(1 \leq i \leq n)$ exits the machine where it is processed, and it remains in the system (shop) during q_i. This is the delivery time (q_i) and it corresponds to a cooling period of the job J_i that leaves a furnace.

Each machine processes at most only one job at the same time. It is worth noting that the completion time, C_i, of a job, J_i, is the date it exits the system (shop). In other terms, $C_i = f_i + q_i$, where f_i is the finishing processing date of job J_i in a machine. Its objective is to find a feasible schedule that minimizes the maximum completion time, C_{max} (makespan), of all jobs, which is expressed as follows:

$$C_{max} = \max_{1 \leq i \leq n} (C_i).$$

The scheduling is performed under the following assumptions:

- All machines are available for the processing of the jobs at time zero and the entire time horizon.
- A job is assigned to exactly one machine at the same time.
- Preemption during the processing of a job is not allowed.
- The processing times, p_i; the release dates, r_i; and the delivery times, q_i, are assumed to be deterministic and integral $(1 \leq i \leq n)$.

According to the three-field notation [47], this problem is denoted as $P_m, NI|r_j, q_j|C_{max}$, where *NI* indicates the non-idle machine time constraint.

In the following, an illustrative example is given.

Example 1. *Consider two furnaces that must heat different pieces (components) to a certain temperature. The pieces arrive at the system on different dates (release dates). The pieces have a delivery time that must elapse between completion processing in a furnace and exiting the system (shop). This corresponds to a cooling period of the processed piece. Maintaining a furnace's required temperature while it is idle is an energy-consuming period without performing any task. This is a waste of energy and consequently an environmental issue. Furthermore, the idle period adds an extra energy cost. This can be avoided by not allowing the idle machine times between consecutive jobs. This could be seen as a green practice.*

The data for the given example are presented in Table 1.

Table 1. Data for example 1.

i	r_i	p_i	q_i
1	2	6	3
2	8	7	2
3	5	3	4
4	3	3	16
5	7	9	6

The example consists of five jobs $(n = 5)$ and two machines $(m = 2)$, where the input parameters are listed in Table 1. Figure 1 shows an optimal solution for Example 1.

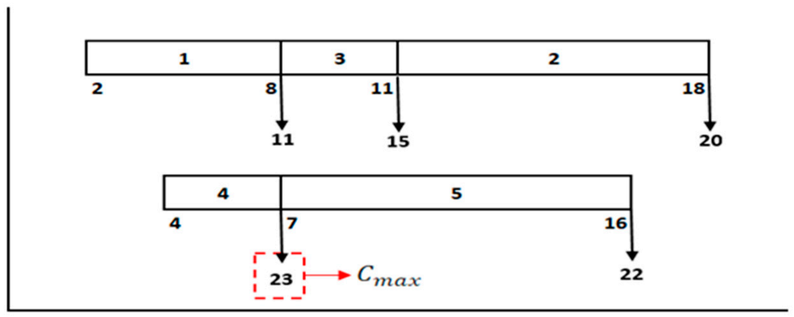


Figure 1. Gant chart of an optimal schedule for the problem $P_m, NI/r_i, q_i/C_{max}$.

The date of the finishing processing for job 1 is 8; this is the date when job 1 exits machine 1. The delivery time (cooling duration) of job 1 is 3. Then, job 1 leaves the system at 8 + 3 = 11; this is the completion time of job 1.

It is worth noting that in the previous solution (Figure 1), job 4 can start at its release date, $r_4 = 3$, but if so, the no-idle machine constraint is not satisfied, and an idle time appears from time 6 to 7. Thus, job 4 is constrained to start later at time 4.

3.2. Problem Properties

In this subsection, several useful and relevant proprieties of the studied problem are presented.

3.2.1. Complexity

Proposition 1. *The problem $P_m, NI|r_j, q_j|C_{max}$ is NP-hard in the strong sense.*

Proof of Proposition 1. Indeed, the problem $P_m|r_j, q_j|C_{max}$, which is a relaxation of $P_m, NI|r_j, q_j|C_{max}$, is NP-hard in the strong sense [48]. □

3.2.2. Symmetry

Remark 1. *The $P_m, NI|r_j, q_j|C_{max}$ is symmetric in the following sense. The problems $P_m, NI|r_j, q_j|C_{max}$ and $P_m, NI|q_j, r_j|C_{max}$ have the same optimal solutions, which means that the roles of the release dates and the delivery time are symmetric.*

Proof of Remark 1. Indeed, if UB is an upper bound and t is the timeline, then considering the new timeline $UB - t$ allows us to retrieve the same schedules for both problems. In particular, this holds for the optimal schedules. □

An immediate consequence of the last remark (Remark 1) is to systematically investigate the original problem $(P_m, NI|r_j, q_j|C_{max})$ and its symmetric $(P_m, NI|q_j, r_j|C_{max})$, while running the algorithms, in order to improve the obtained solution. This is applicable for the lower bounds and the metaheuristics that are proposed later in this work.

In the following, the original problem $(P_m, NI|r_j, q_j|C_{max})$ is referred to as the Forward problem, and the symmetric problem $(P_m, NI|q_j, r_j|C_{max})$ is referred to as the Backward problem.

3.2.3. Relationship between the Problems $P_m | r_j,q_j | C_{max}$ and $P_m,NI | r_j,q_j | C_{max}$

Proposition 2. *If LB is a lower bound for the problem $P_m|r_j, q_j|C_{max}$, then LB is also a lower bound for the problem $P_m, NI|r_j, q_j|C_{max}$.*

Proof of Proposition 2. An optimal schedule, S^*_{NI}, for $P_m, NI|r_j, q_j|C_{max}$ with an optimal value, C^*_{NI}, is a feasible schedule for $P_m|r_j, q_j|C_{max}$. If C^* is the optimal value of $P_m|r_j, q_j|C_{max}$, then $C^* \leq C^*_{NI}$. Consequently, if LB is a lower bound for $P_m|r_j, q_j|C_{max}$, then $LB \leq C^* \leq C^*_{NI}$. □

Corollary 1. If LB denotes the optimal solution of $P_m|r_j, q_j|C_{max}$, then LB is a lower bound of $P_m, NI|r_j, q_j|C_{max}$.

Proof of Corollary 1. An optimal solution of $P_m|r_j, q_j|C_{max}$ is also a lower bound for $P_m|r_j, q_j|C_{max}$. Based on Proposition 2, this optimal solution is a lower bound for $P_m, NI|r_j, q_j|C_{max}$. □

Remark 2. *The lower bound LB (Corollary 1) is the optimal solution of the $P_m|r_j, q_j|C_{max}$. In this study, the Branch and Bound (B&B) algorithm developed in [48] was used to optimally solve the problem $P_m|r_j, q_j|C_{max}$ due to its efficiency. In addition, $P_m|r_j, q_j|C_{max}$ is NP-hard, and the (B&B) might fail to solve it optimally within a prefixed time limit (300 s). In this case, the best obtained lower bound while running the (B&B) is considered as the lower bound for the current studied problem.*

In the following, the obtained lower bound from Corollary 1 and Remark 2 will be adopted as a lower bound for the problem $P_m, NI|r_j, q_j|C_{max}$, and will be denoted LB.

4. Mixed Integer Linear Formulation

In this section, a mixed integer linear program (MILP) is proposed. The MILP model was developed to determine the optimal solution for this problem and was solved by using the programming software IBM ILOG CPLEX (Academic).

The main issue while developing the mixed integer linear program is the integration of the no-idle constraint. To this aim, a dummy job denoted by J_0 was included within the set of jobs. The processing time, release date, and delivery time of this job were set to 0, respectively. This dummy job is useful to serve as the predecessor for all jobs.

The notations for the developed MILP are presented in Table 2.

Table 2. MIPL notations.

Indices:	
i,	index of job, where $\mathrm{i,j} = 1, 2, \ldots, \mathrm{n}$.
k	index of machine, where $\mathrm{k} = 1, 2, \ldots, \mathrm{m}$.
Parameters:	
$\mathrm{p_i}$	Processing time of job i.
$\mathrm{r_i}$	Release date of job i
$\mathrm{q_i}$	Delivery time of job i
M	A large number
Decision variables	
$\mathrm{x_{ik}}$	A binary decision variable equal to 1 if job i is processed in machine k, otherwise 0.
$\mathrm{y_{ijk}}$	A binary decision variable equal to 1 if job j is processed immediately after job i and is being processed on machine k; otherwise 0.
$\mathrm{C_{max}}$	Maximum completion time
$\mathrm{s_j}$	Starting time of the job j.
$\mathrm{f_j}$	Completion time of the job j.

The MILP model can be formulated as follows.

$$\boldsymbol{min\ C_{max}} \tag{1}$$

Subject to

$$f_i + q_i - C_{max} \leq 0, \quad i = 1, 2, \ldots, n \tag{2}$$

$$\sum_{k=1}^{m} x_{ik} = 1, \quad i = 1, 2, \ldots, n \tag{3}$$

$$\sum_{(i=0, i \neq j)}^{n} y_{ijk} = x_{jk}, \quad j = 1, 2, \ldots, n;\ k = 1, 2, \ldots, m \tag{4}$$

$$\sum_{(j=1, i \neq j)}^{n} y_{ijk} \leq x_{ik}, \quad i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, m \tag{5}$$

$$\sum_{(j=1)}^{n} y_{0jk} = 1, \quad k = 1, 2, \ldots, m \tag{6}$$

$$f_i = s_i + \sum_{k=1}^{m} p_i\, x_{ik}, \quad i = 1, 2, \ldots, n \tag{7}$$

$$f_0 = 0 \tag{8}$$

$$f_i - s_j + M\left(\sum_{k=1}^{m} y_{ijk} - 1\right) \leq 0, \quad i = 0, 1, \ldots, n;\ and\ j = 1, 2, \ldots, n \tag{9}$$

$$r_i - s_i \leq 0, \quad i = 1, 2, \ldots, n \tag{10}$$

$$f_j - s_i - p_i - p_j + M\left(y_{ijk} - 1\right) \leq 0, \quad i, j = 1, 2, \ldots, n;\ and\ k = 1, 2, \ldots, m \tag{11}$$

$$x_{ik} \in \{0,\ 1\}, \quad i = 1, 2, \ldots, n;\ and\ k = 1, 2, \ldots, m \tag{12}$$

$$y_{ijk} \in \{0,\ 1\}, \quad i = 1, 2, \ldots, n;\ j = 0, 1, \ldots, n;\ k = 1, 2, \ldots, m \tag{13}$$

$$s_i,\ f_i,\ C_{max} \geq 0, \quad i = 1, 2, \ldots, n \tag{14}$$

- Equation (1) is the objective function, which minimizes the maximum completion time (makespan).
- Constraint (2) proposes that the completion time $f_i + q_i$ of each job $i \in J$ is less than the maximum completion time C_{max}.
- Constraint (3) proposes that each job should be assigned to exactly one machine.
- Constraint (4) proposes that each processed job, $i \in J$, in a machine has a unique immediate predecessor. It is worth mentioning that the first processed job in each machine has as immediate predecessor, the dummy job J_0.
- Each processed job has at most one immediate successor job; this is the meaning of constraint (5).
- Constraints (6) proposes that the first processed job in each machine has as immediate predecessor, the dummy job J_0.
- Equation (7) expresses the relationship between the starting time, s_i, and the completion processing time, f_i, for each job $i \in J$.
- Equation (8) stipulates that the dummy job, J_0, finishes processing at time 0. Therefore, it precedes all other jobs.
- Constraint (9) expresses the precedence constraint between two consecutive jobs. These constraints eliminate any overlap between the consecutive jobs.
- Constraint (10) causes each job $i \in J$ to start processing after its release date, r_i.
- Constraint (11) ensures that the difference between the start time of a job and the end of its predecessor job is not larger than the total processing time (satisfaction of the no-idle time constraint). Indeed, if job j is processed immediately after job i on machine k, then $y_{ijk} = 1$. Consequently, constraint (11) proposes that $f_j \leq (s_i + p_i) + p_j$. Based on (7), we have $s_i + p_i = f_i$, then $f_j \leq f_i + p_j$. Constraint (9) proposes that $f_i \leq s_j$ and $f_j \leq f_i + p_j \leq s_j + p_j = f_j$. Thus, $f_i + p_j = f_j$, and there is no idle time between

jobs *i* and *j*. These are the constraints expressing the green scheduling aspect, since the idle machine times are eliminated, which allows the idle consumed energy to be eliminated. This is the main difference with a classical scheduling problem (with idle machine times).

- Constraint sets (12) and (13) enable integrality for binary decision variables. Finally, constraint (14) imposes non-negativity for real decision variables.

5. Metaheuristics Procedures

The studied problem is NP-hard in the strong sense. In order to provide a near optimal solution, several metaheuristics were investigated and adapted. More precisely, two metaheuristics were developed in order to solve the studied scheduling problem. These metaheuristics are the genetic algorithm (GA) and the simulated annealing algorithm (SA). This choice is justified by a study of the most recent literature, for the green parallel machine scheduling problems using metaheuristics. According to this literature [49–55], the genetic algorithm and simulated annealing are successful methods for solving parallel machine scheduling problems. In this context, three variants of GA were developed. These variants use three different crossover operators. The algorithm parameter tuning is included in the experimental section.

5.1. Genetic Algorithms

Genetic algorithms (*GA*) were developed first by John Holland in [56]. GAs attempt to simulate the process of natural evolution through (genetic) selection by following the principles of natural evolution in a given environment. Natural evolution does not act directly on living beings, but it operates on the chromosomes' DNA. However, the used vocabulary in GAs is similar to that in natural genetics, but the natural genetics processes are much more complicated than the processes of the GA models.

The terminology used in GAs depends on natural genetic processes: chromosomes are the elements from which they are made (individuals). These chromosomes are grouped into populations, and a combination of chromosomes is considered the reproductive stage. This is performed utilizing a mutation operator and/or a crossing operator. Other concepts are specific to the GA field, such as the quality index (fitness), also called the performance index, which is a measure used to classify chromosomes. The same goes for the evaluation function, representing the theoretical formula for calculating and finding the quality index of a chromosome.

For a given optimization problem, an individual represents a feasible solution in the solution space. Every individual associated with the value of the objective function is optimized. The iterative search generates populations of individuals on which we apply selection, crossing, and mutation processes. The selection aims to promote the best elements of the population for the criterion considered (the best suited), mutation, and crossing, which ensure an exploration of the solution space. Figure 2 shows the simplified iterative operation of GAs.

To create GAs and solve problems effectively, it is important to identify how to represent the solutions (chromosome representation), define an evaluation function, develop the different operators, and determine the parameters, such as the choice of stopping criteria and the probability of an operator's application. Representation, fitness, initial population, genetic operators, and standard stopping rules are discussed in detail below.

To implement a GA for the problem under study, the first step is to form a good representation of the problem's information in the form of a chromosome. This chromosome's representation must be complete, considering all the possible solutions to the problem, which can be codified using this representation. Additionally, all the adjustments on this chromosome must correspond to feasible solutions (principle of validity).

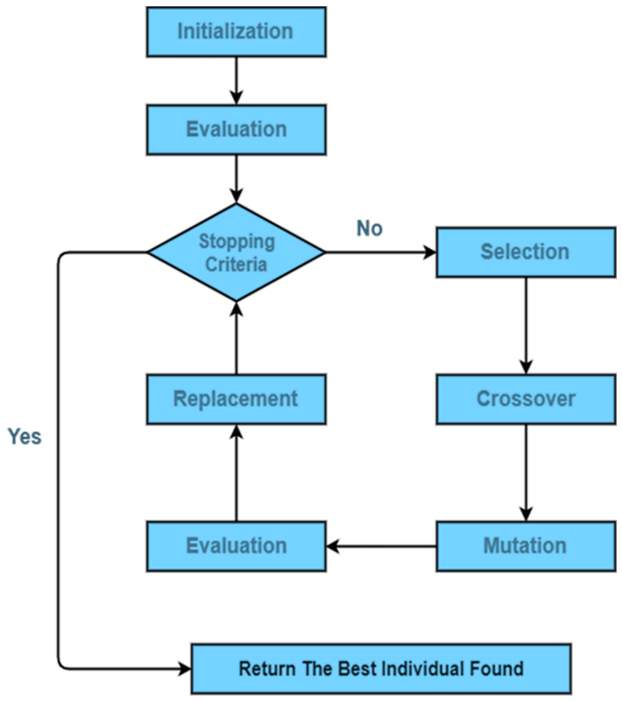


Figure 2. Simplified iterative operation of GAs.

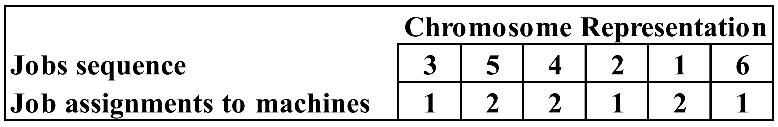
5.1.1. Chromosome Representation

For the problem considered in this paper, the chromosome's representation should simultaneously reflect two main characteristics:

- The job assignments to machines;
- The sequence of assigned jobs in each machine.

In this study, a chromosome is represented by a permutation of jobs. The decoding consists of assigning the first unscheduled job in the permutation to the most available machine, with the elimination of the idle time by right shifting the jobs until no idle time is detected.

Example 2. *The chromosome when $n = 6$ and $m = 2$ is presented in Figure 3.*



	Chromosome Representation					
Jobs sequence	3	5	4	2	1	6
Job assignments to machines	1	2	2	1	2	1

Figure 3. Sample of chromosome representation.

From the chromosome representation, the sequence on machine 1 is $3 - 2 - 6$, and the sequence on machine 2 is $5 - 4 - 1$.

5.1.2. Initial Population

The GA's initial population represents the algorithm's starting population, which can strongly condition the speed of an algorithm. If the optimum position in the solution space is completely unknown, it is natural to randomly generate individuals by making uniform draws of the solution space. On the other hand, if a priori information on the problem is available, it seems to create individuals in a particular sub-domain to accelerate convergence. The prior knowledge from our research literature could be utilized for some dispatching rules or simple heuristics, such as the earliest release date, Jackson's algorithm, or the shortest processing time. In this study, the initial population was randomly generated.

5.1.3. Selection Operator

Unlike other optimization techniques, genetic algorithms do not require any particular hypothesis on the objective function's regularity. In particular, the genetic algorithm does not use its successive derivatives, making its field of application very broad. No continuity assumption is required either. The selection operator chooses the most suitable individuals to enable the closest solution population to converge towards the global optimum. However, in the literature, many selection techniques are more or less adapted to problems they deal with. In this study, the roulette wheel selection rule was used [57].

5.1.4. Reproduction

Reproduction means the cloning of an individual without modification, which will pass to the next generation. In this way, reproduction is an alternative genetic operator to crossing and mutation, since they modify the individuals that pass into the next generation. The purpose of reproduction is to keep individuals with high fitness of the present age in the next generation.

5.1.5. Crossover Operator

Crossover aims to enrich the diversity of the population by manipulating the structure of the chromosomes [58]. Conventionally, crosses are considered with two parents and generate two children. The selection process chooses both parents. Crossbreeding allows innovation (children are different from their parents) and is based on the idea that two successful parents will produce better children. The crossover rate p_c $(p\_c \in [0,1])$ represents the proportion of parents on which a crossover operator will act. There are several crossover operators in the literature regarding the creation of new child chromosomes with the best fitness value. In the following, the most used crossover operators are as follows: (1) the block order crossover operator (BOX) [2], (2) linear order crossover operator (LOX) [3], and (3) position-based crossover operator (POX) [4]. In this study, three variants of GA are proposed. These variants use crossover operators (BOX), (LOX), and (POX), respectively. These metaheuristics are denoted as GA_{BOX}, GA_{LOX}, and GA_{POX}, respectively. All three variants of GA share the same operators, except the crossover ones.

5.1.6. Mutation

The mutation is considered a primary operator, providing a small randomness element in individuals in the population [59]. Although it is recognized that the crossing operator is responsible for searching for the space of possible solutions, the mutation operator is responsible for increasing or reducing the search space within the genetic algorithm as well as for promoting the genetic variability of the individuals in the population. The probability (or ratio) p_m defines the probability of mutating each element (gene) of representation. There are several techniques for applying the mutation to individuals in a population, but the most commonly used is to mutate the percentage of total genes in the population.

5.1.7. Replacement Strategies

The replacement phase concerns the survivor selection of both the parent and offspring populations. When the population's size is constant, it allows individuals to be withdrawn according to a given selection strategy. Elitism always consists of selecting the best individuals from the parents and the offspring. This method leads to rapid convergence, and a premature convergence could occur. Sometimes, selecting bad (in terms of fitness) individuals is necessary to avoid the sampling error problem. These replacement strategies may be stochastic or deterministic.

5.2. Simulated Annealing

The simulated annealing (SA) algorithm was firstly proposed in [60] to deal with highly non-linear problems, and afterward, the authors suggested it for solving combinatorial optimization problems. Since then, SA has had a significant impact on the field of metaheuristic research for its simplicity and efficiency in solving combinatorial optimization problems by escaping local optima. It has also been extensively studied to deal with continuous optimization problems also applied to numerous other areas. It originates from the fields of material science and physics [5], where the SA algorithm simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state (steady frozen state).

The SA is one of the popular metaheuristics successfully applied to various combinatorial optimization problems. SA is a memoryless algorithm because the algorithm does not use any information gathered during the search. From an initial solution, SA proceeds in several iterations.

The SA improves a solution by iteratively moving the current solution s to a neighborhood solution $\acute{s}$, generated randomly. If $\acute{s}$ is better than s, then the movement from s to $\acute{s}$ is accepted, i.e., s is replaced by $\acute{s}$. If $\acute{s}$ is worse than s, it is accepted with a probability of $e^{-\Delta E/T}$, called an uphill move, where ΔE represents the difference between the objective function values of s and $\acute{s}$, and T is a parameter called the temperature. As the algorithm progresses, the probability that such moves are accepted decreases. The distribution of the probability equation is as follows:

$$P(\Delta E, T) = e^{-\Delta E/T}$$

When the temperature increases, the probability of accepting the worst move also increases. At a given temperature, the lower the objective function's increase is, the more significant the likelihood of accepting the move is. T is initially set to $T_0 = T_{max}$, and is decreased after every iteration. The algorithm (Algorithm 1) is terminated if temperature T reaches $T_f = T_{min}$. It is worth noting that T_{max} and T_{min} are the maximum and minimum temperatures, respectively. The best solution found at the beginning of the search is stored in addition to the current solution.

Algorithm 1 SA algorithm

```
        **Input**: Cooling schedule.
        $s = s_0$; %Generation of the initial solution
        $T = T_{max}$; % Starting temperature
        **Repeat**
**Repeat** % At a fixed temperature
            % Generate a random neighbor $s'$
            $\Delta E = f(s') - f(s)$;
            **If** $\Delta E \leq 0$ then $s = s'$ % Accept the neighbor solution
            **Else** accept $s'$ with a probability $e^{-\Delta E/T}$
            **Until** Equilibrium condition
            %, e.g., a given number of iterations executed at each temperature T
            $T$ = g(T); %Temperature update
        **Until** Stopping criteria satisfied %, e.g., $T < T_{min}$
        **Output**: Best solution found
```

A few parameters control the search's progress, which is the temperature and the number of iterations performed at each temperature. The main elements of SA can be summarized as follows:

- The acceptance probability function: the main element of SA that enables non-improving neighbors to be selected.
- The cooling schedule: tis defines the temperature at each step of the algorithm. It plays an essential role in the efficiency and effectiveness of the algorithm.

Regarding the stopping condition, the theory suggests a final temperature equal to 0. In practice, one can stop the search when the probability of accepting a move is negligible. The following stopping criteria may be utilized:

- Reaching a final temperature T_F.
- Achieving a predetermined number of iterations without improvement.
- The objective function reaches a pre-specified threshold value (e.g., lower bound).
- A predetermined number of evaluations.

6. Experimental Study and Results

This section evaluates the performance of the proposed MILP, GAs, and SA, and compares the proposed procedures to the existing ones. Computational experiments were carried out with 4000 instances for the proposed algorithms. The proposed algorithms were coded using C++ and the MILP using CPLEX software. The computational experiments were undertaken on an MSI computer with the following specifications: processor: Intel (R) Core™ i7-7700 HQ CPU at 2.8 GHz; RAM: 8 GB.

6.1. Data Set Generation

The test problems were generated as in previous studies [45]. Two classes of instances were generated (Class A and Class B) with different problem sizes of jobs (n = 10, 20, 40, 50, and 200) (m = 2,3,5,8). Parameters for Class A and Class B are presented as follows.

6.1.1. Class A

- The processing times p_i were randomly and uniformly generated in $[1, 10]$.
- The release dates, r_i, and delivery times, q_i, were randomly and uniformly generated in $\left[1, \left(\frac{n*k}{m}\right)\right]$, where $(k = 1, 3, 5, 7, 10, 13, 17, 22, 27, 33)$.

6.1.2. Class B

- The processing times, p_i, were randomly and uniformly generated in $[1, n]$.
- The release dates, r_i, were randomly and uniformly generated in $[1, n]$.
- The delivery times, q_i, were randomly and uniformly generated in $\left[1, \left(\frac{n*k}{m}\right)\right]$, where $(k = 1, 3, 5, 7, 10, 13, 17, 22, 27, 33)$.

Combinations of different parameters resulted in a total of 4000 test problems.

6.2. Parameters Tuning

In this section, the parameter design of GAs and SA is investigated and tuned. Different combinations of parameters returned different results for the metaheuristic algorithms, which means the parameter values used in each algorithm affect their performance.

To identify each design parameter's proper settings, a pilot run was conducted based on screening and the literature. The Taguchi design of L9 [32] was used to study the effect of the parameters of the proposed algorithm on the makespan, C_{max}, and the best settings for each proposed metaheuristic parameters were determined. The detailed results for the tuned variables are given in Table 3 and Figures A1–A6 (Appendix A).

More precisely, the GA parameters that should be determined are as follows:

- Population size;
- Crossover rate;
- Mutation rate;
- Stopping condition.

The SA parameters that should be determined before the experimental study are as follows:

- Temperature;
- Maximum number of iterations: MAX_ITER;
- Temperature reduction factor: ALPHA;

- Number of function evaluations before temperature reduction: NT factor.

Table 3. Summary of the main parameters and levels for GAs.

Population Size	Crossover Rate	Mutation Rate	Stopping Condition	*n* = 10	20	40	50	200
40	0.4	0.2	100	1.06	8.21	18.69	21.42	11.28
40	0.75	0.5	1000	0.56	5.38	12.51	16.84	8.85
40	0.95	0.9	10,000	0.19	3.67	9.37	13.99	8.03
80	0.4	0.5	10,000	0.28	2.33	7.14	10.96	6.71
80	0.75	0.9	100	0.46	6.63	13.96	17.45	9.73
80	0.95	0.2	1000	0.19	4.06	9.97	18.06	8.45
120	0.4	0.9	1000	0.10	2.86	8.72	11.19	7.33
120	0.75	0.2	10,000	0.31	2.31	6.52	9.32	6.19
120	0.95	0.5	100	0.46	6.33	14.02	16.10	9.99

The determination of these parameters requires a preliminary experimental phase, performed in a reduced number of instances, and on a given set of values for each parameter (Table 4). For example, the population size parameter belongs to the set $\{40, 80, 120\}$, and one of these values has to be selected. The design of this experimentation is performed following the Taguchi design of L9 [32].

Table 4. Summary of the main parameters and levels for GAs and SA.

	Parameter	Considered Values	Selected Values
GA	Population Size	40, 80, 120	120
	Crossover Rate (Pc)	0.4, 0.75, 0.95	0.95
	Mutation Rate (Pm)	0.2, 0.5, 0.9	0.9
	Stopping Condition	Number of Evaluations (100, 1000, 10K) or *LB*	10K or *LB*
SA	Temperature	5, 5 Thousand, 5 Million	5
	MAX_ITER (maximum number of iterations)	100, 1000, 10,000	10,000
	ALPHA (temperature reduction factor)	0.5, 0.75, 0.95	0.5
	NT factor (number of function evaluations before temperature reduction)	10, 75, 150	10

Figures A1–A5 present the average relative percent deviation (ARPD) for each parameter and for each number of jobs (n). Each figure contains four curves, where each one represents a specific parameter. If, for example, we focus on the population size parameter selection, we observe that for all numbers of jobs, the minimum $ARPD$ is reached for $Population\ Size = 120$. Therefore, the adopted value of the population size parameter is 120. This reasoning is valid for the rest of the parameters. Figure A6 is reserved for the SA parameter selection, and the same reasoning as for GA holds.

The summary of the main parameters and their levels regarding GAs and SA are shown in Table 4.

6.3. Results and Discussions

In this subsection, the proposed procedures (GAs, SA, MILP and *LB*) are assessed. First, a pairwise comparison between the metaheuristics was performed using the average relative percent deviation ($ARPD$) and the average computation time ($Time$). Then, the best metaheuristic was selected, and its absolute efficiency was evaluated throughout the average relative gap using the lower bound (LB). In addition, the best obtained metaheuristic was compared to the MILP formulation. Finally, a comparison of the best metaheuristic with existing algorithms was carried out.

6.3.1. Metaheuristics Pairwise Comparison

First, the following should be noted:

- GA_{BOX} denotes the genetic algorithm based on the block order crossover operator.
- GA_{LOX} denotes the genetic algorithm based on a linear order crossover operator.
- GA_{POX} denotes the genetic algorithm based on a position-based crossover operator.
- SA denotes the simulated annealing algorithm.

For instance, we adopted the following notations and definitions:

- C_{BOX} denotes the makespan obtained via GA_{BOX}.
- C_{LOX} denotes the makespan obtained via GA_{LOX}.
- C_{POX} denotes the makespan obtained via GA_{POX}.
- C_{SA} denotes the makespan obtained via SA.
- $C_{best} = \min(C_{BOX}, C_{LOX}, C_{POX}, C_{SA})$.
- For a given algorithm, $H \in \{GA_{BOX}, GA_{LOX}, GA_{POX}, SA\}$, the relative percent deviation (with reference to C_{best}) is defined as follows:

$$RPD = \frac{C_H - C_{best}}{C_{best}} \times 100$$

- For a subset of instances, $ARPD$ is the average RPD.

The three variants of GA and the SA algorithm were compared using the $ARPD$ and the average consumed time, $Time(s)$, running the algorithm. The detailed results are presented in Tables 5 and 6 and Figures 4 and 5.

Table 5. Proposed metaheuristics results for Class A.

m	*n*	***ARPD***				***Time(s)***			
		GA_{BOX}	GA_{LOX}	GA_{POX}	SA	GA_{LOX}	GA_{POX}	GA_{POX}	SA
2	10	0.1%	0.0%	0.0%	0.4%	1.37	1.24	1.49	0.11
	20	0.2%	0.0%	0.0%	2.2%	2.23	1.80	2.25	0.11
	40	1.4%	0.2%	0.0%	4.7%	5.13	2.92	3.74	0.11
	50	3.1%	0.6%	0.0%	6.1%	6.75	3.55	4.43	0.12
	200	14.9%	5.3%	0.0%	11.7%	32.18	20.94	18.72	0.15
3	10	0.0%	0.0%	0.0%	0.3%	1.39	1.28	1.52	0.11
	20	0.3%	0.0%	0.0%	2.2%	2.39	1.87	2.31	0.11
	40	2.8%	0.3%	0.0%	4.9%	4.98	3.09	3.89	0.11
	50	3.7%	0.7%	0.0%	5.4%	6.62	3.76	4.63	0.12
	200	13.8%	4.8%	0.0%	10.4%	64.61	21.76	19.52	0.15
5	10	0.2%	0.0%	0.0%	0.3%	1.31	1.25	1.49	0.11
	20	1.4%	0.1%	0.0%	2.4%	2.37	1.85	2.30	0.11
	40	4.5%	0.5%	0.0%	4.4%	5.30	3.15	3.91	0.11
	50	5.8%	0.5%	0.0%	5.1%	7.08	3.85	4.65	0.12
	200	13.3%	4.1%	0.0%	9.4%	61.61	22.17	19.93	0.15
8	10	0.1%	0.0%	0.0%	0.0%	1.23	1.21	1.44	0.11
	20	2.5%	0.2%	0.0%	2.3%	2.34	1.81	2.28	0.11
	40	3.7%	0.3%	0.0%	3.0%	5.26	3.11	3.86	0.12
	50	5.1%	0.5%	0.0%	4.0%	6.97	3.85	4.66	0.12
	200	12.2%	3.3%	0.0%	8.7%	60.35	22.78	20.46	0.16
Average		4.5%	1.1%	0.0%	4.4%	14.07	6.36	6.37	0.12

Table 6. Proposed metaheuristics results of Class B.

m	n	*ARPD*				*Time(s)*			
		GA_{BOX}	GA_{LOX}	GA_{POX}	SA	GA_{BOX}	GA_{LOX}	GA_{POX}	SA
2	10	5.8%	1.0%	0.0%	0.7%	1.11	1.27	1.36	0.37
	20	16.0%	1.4%	0.0%	4.3%	1.86	1.62	2.01	0.37
	40	26.1%	6.9%	0.0%	11.5%	5.48	2.74	3.77	0.60
	50	26.3%	10.6%	0.0%	13.3%	7.09	3.42	4.26	0.83
	200	7.9%	2.5%	0.1%	3.9%	54.82	21.89	19.43	0.93
3	10	5.2%	0.7%	0.0%	0.8%	1.11	1.39	1.45	0.21
	20	18.0%	2.2%	0.0%	6.4%	1.89	1.70	2.28	0.32
	40	25.8%	7.1%	0.0%	12.6%	5.53	2.82	3.85	0.56
	50	26.3%	9.0%	0.0%	14.5%	7.23	3.50	4.33	0.82
	200	10.0%	2.9%	1.1%	5.9%	50.59	22.02	20.93	0.91
5	10	1.6%	0.0%	0.0%	0.1%	1.13	1.33	1.47	0.59
	20	17.3%	2.3%	0.0%	5.9%	1.85	1.72	2.36	0.53
	40	28.8%	8.8%	0.0%	13.4%	5.49	2.84	3.59	0.71
	50	27.9%	9.4%	0.0%	14.0%	7.43	3.48	4.29	0.87
	200	12.9%	2.7%	2.5%	7.0%	50.95	21.53	19.20	0.68
8	10	0.3%	0.0%	0.0%	0.0%	1.12	1.21	1.50	0.78
	20	14.1%	1.3%	0.0%	3.5%	1.86	1.72	2.23	0.74
	40	28.2%	6.5%	0.0%	11.0%	5.42	2.82	3.58	0.79
	50	29.1%	8.5%	0.0%	12.9%	7.45	3.44	4.26	0.78
	200	13.1%	2.2%	0.8%	4.2%	52.32	21.14	18.79	0.96
Average		17.0%	4.3%	0.2%	7.3%	13.59	6.18	6.25	0.67

Based on Table 5 and Figure 4, the GA_{POX} metaheuristic largely outperforms the other metaheuristics for Class A, with an average of $ARPD = 0\%$. In addition, the average consumed computation time is satisfactory and $Time = 6.37$ s. The SA presents the best average time with $Time = 0.12$ s. According to Table 6 and Figure 5, the same behavior is detected for Class B. Indeed, the GA_{POX} algorithm is the leading metaheuristic with $ARPD = 0.2\%$. More precisely, $ARPD = 0\%$, except for $n = 200$, where $ARPD \leq 2.5\%$.

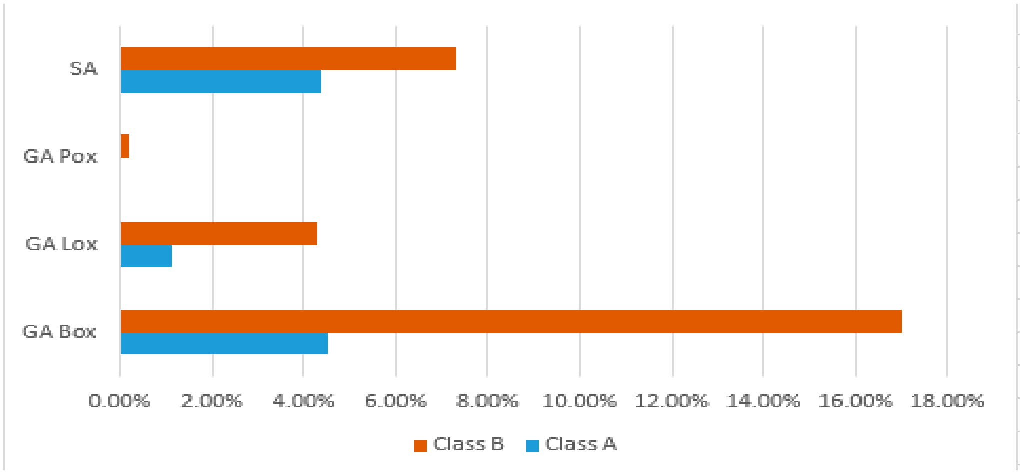


Figure 4. *ARPD* of different proposed algorithms with a random initial population.

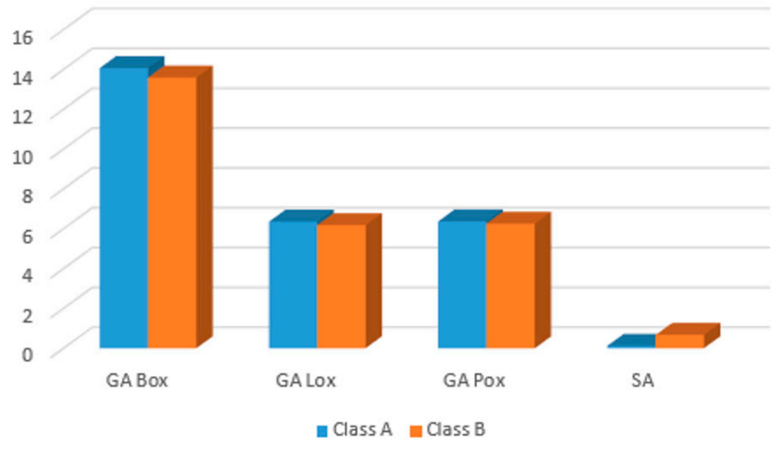


Figure 5. Average consumed time of different proposed algorithms with a random initial population.

For the two classes (A and B), the worst performance is detected for the GA_{BOX} metaheuristic in terms of $ARPD$ and $Time$. Indeed, for this metaheuristic, the $ARPD = 4.5\%$ and $Time = 14.07$ s for Class A, and $ARPD = 17.0\%$ and $Time = 13.59$ s for Class B. The two remaining metaheuristics (GA_{LOX},SA) behave almost in the same way.

Except the GA_{POX}, the other metaheuristics are very sensitive to n and m. Indeed, for these metaheuristics, the $ARPD$ and $Time$ increase dramatically as n and m increase. Therefore, the metaheuristic GA_{POX} is selected as the best metaheuristic among the four proposed ones.

6.3.2. Performance of the Metaheuristic GA_{POX}

Based on the previous subsection, GA_{POX} presents the best performance compared to the others. In this subsection, and based on the proposed lower bound (LB) (in Section 3.2.3; Remark 2), a relative percent deviation (RPD) between LB and the obtained C_{max} was used to assess the performance of the proposed algorithm. The relative percent deviation is expressed as follows:

$$RPD = \frac{C_{GA_{POX}} - LB}{LB} \times 100$$

The RPD provides an upper bound on the relative absolute distance between the makespan of the metaheuristic GA_{POX} and the optimal solution (which is unknown). Indeed, if C^*_{max} denotes the optimal value of the makespan, then $LB \le C^*_{max} \le C_{GA_{POX}}$. Therefore,

$$RPD = \frac{C_{GA_{POX}} - LB}{LB} \times 100 \ge \frac{C_{GA_{POX}} - C^*_{max}}{C^*_{max}} \times 100.$$

Therefore, the RPD allows the solution quality to be evaluated. The related results are detailed in Table 7. In the latter table, the used key performance measures are the average, the minimum, and the maximum RPD, which are denoted, respectively, as Average, Min, and Max.

Based on Table 7, the average relative deviation is $ARPD = 0.06\%$ for Class A and $ARPD = 0.11\%$ for Class B. This is strong evidence that the proposed metaheuristic performs well, since the $ARPD$ is very low. This means that the provided solution is too close to the optimal solution. In addition, the maximum $ARPD$ for Classes A and B are, respectively $ARPD = 14.71\%$ and $ARPD = 15.60\%$. The maximum $ARPD$ is obtained for both $n = 40$ and $m = 8$.

Table 7. $ARPD$ computational results for GA based on POX.

m	n	***ARPD*** **(*Class A*)**			***ARPD*** **(*Class B*)**		
		Average	**Min**	**Max**	**Average**	**Min**	**Max**
2	10	0.04%	0.00%	4.35%	0.00%	0.00%	0.00%
	20	0.03%	0.00%	1.61%	0.00%	0.00%	0.00%
	40	0.00%	0.00%	0.00%	0.00%	0.00%	0.19%
	50	0.10%	0.00%	8.16%	0.09%	0.00%	3.53%
	200	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%
3	10	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	20	0.02%	0.00%	2.17%	0.01%	0.00%	1.18%
	40	0.02%	0.00%	1.28%	0.06%	0.00%	1.40%
	50	0.00%	0.00%	0.33%	0.06%	0.00%	3.92%
	200	0.00%	0.00%	0.00%	0.00%	0.00%	0.03%
5	10	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	20	0.17%	0.00%	6.25%	0.12%	0.00%	4.92%
	40	0.29%	0.00%	10.42%	0.22%	0.00%	3.47%
	50	0.12%	0.00%	6.25%	0.19%	0.00%	3.94%
	200	0.00%	0.00%	0.43%	0.03%	0.00%	0.20%
8	10	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	20	0.20%	0.00%	10.00%	0.00%	0.00%	0.00%
	40	0.15%	0.00%	14.71%	0.63%	0.00%	15.60%
	50	0.02%	0.00%	2.38%	0.56%	0.00%	3.75%
	200	0.00%	0.00%	0.00%	0.17%	0.00%	0.67%
Average		0.06%	0.00%	3.42%	0.11%	0.00%	2.14%

The minimum $ARPD$ for each m is observed for $n = 10$ and $n = 200$. For each number of machines, m, the $ARPD$ presents a maximum for a certain number of jobs n; for example, for $m = 8$, the maximum is reached for $n = 20$ (Class A). This maximum depends on m.

6.3.3. Comparison of MILP with GA_{POX}

Since the problem is NP-hard in the strong sense, developing an exact method (MILP) to solve small instances for the studied problem could be useful. Indeed, the solution for MILP can provide a good basis for the experimental assessment of the developed approximation approach GA_{POX}. In this study, the MILP (as expected) is able to optimally solve only small-sized instances within an acceptable computational time. In order to test the efficiency of the proposed GA_{POX}, the optimal solution C_{MILP} provided by MILP (for $n = 10$, $m = 2$ and 3) was compared to the results obtained by GA_{POX}, throughout the computational time (*Time*) and the relative percent deviation, as follows:

$$RPD = \frac{C_{GA_{POX}} - C_{\text{MILP}}}{C_{\text{MILP}}} \times 100$$

The results are reported in Table 8. Based on this table, we observe that the optimal solution is reached by the GA_{POX} for all the instances within an average time of 0.01 s. Consequently, this represents a preliminary satisfactory result for GA_{POX}. In addition, we observe that the *MILP* procedure is a time-consuming procedure, since for some small-sized instances, the average computation time exceeds 1000 s.

6.3.4. GA_{POX} Metaheuristic Comparison with Existing Methods

To evaluate the performance of GA_{POX}, a comparison with existing methods [45] was performed. The comparison was carried out on the same test problems. The existing methods are a set of four heuristics, namely, $\text{H}_{\text{EP}-\text{EP}}$, $\text{H}_{\text{MS}-\text{MS}}$, $\text{H}_{\text{EP}-\text{MS}}$, and $\text{H}_{\text{MS}-\text{EP}}$. These heuristics are two-phase procedures. In the first phase, a feasible solution is produced, while in the second phase, an improvement procedure is carried out. These heuristics are based on the Modified Schrage algorithm (MS) and/or Exact Procedure (EP) (see [45] for more details). The comparison between the GA_{POX} and the four existing heuristics was performed over $ARPD$ relatively to the lower bound, LB. In addition, the average computation time ($TIME$) was used to complete the comparison. The global results are provided in Table 9.

Table 8. Comparison between *MILP* and GA_{POX} for $n = 10$, $m = 2$, and 3.

Class	m	k	$ARPD$ (GA_{POX}: $MILP$)			Time(s) for MILP			Time(s) for GA_{POX}		
			Average	Max	Min	Average	Max	Min	Average	Max	Min
A	2	1	0.0%	0.0%	0.0%	796.91	1338.23	403.72	0.01	0.02	0.00
		3	0.0%	0.0%	0.0%	70.80	251.40	2.13	0.08	0.38	0.00
		5	0.0%	0.0%	0.0%	2.78	4.61	2.11	0.00	0.02	0.00
		7	0.0%	0.0%	0.0%	2.16	2.24	2.11	0.00	0.00	0.00
		10	0.0%	0.0%	0.0%	2.04	2.18	1.94	0.00	0.02	0.00
	3	1	0.0%	0.0%	0.0%	646.37	1004.51	400.51	0.03	0.11	0.00
		3	0.0%	0.0%	0.0%	13.22	41.31	2.13	0.00	0.02	0.00
		7	0.0%	0.0%	0.0%	2.30	2.99	2.11	0.00	0.02	0.00
		5	0.0%	0.0%	0.0%	2.44	2.89	2.21	0.00	0.02	0.00
		10	0.0%	0.0%	0.0%	2.14	2.23	2.01	0.00	0.00	0.00
B	2	1	0.0%	0.0%	0.0%	542.67	948.22	136.01	0.00	0.02	0.00
		3	0.0%	0.0%	0.0%	141.56	296.54	13.27	0.00	0.02	0.00
		5	0.0%	0.0%	0.0%	73.25	178.50	2.35	0.01	0.02	0.00
		7	0.0%	0.0%	0.0%	43.11	204.29	2.38	0.00	0.02	0.00
		10	0.0%	0.0%	0.0%	2.22	2.33	2.11	0.00	0.00	0.00
	3	1	0.0%	0.0%	0.0%	208.90	449.43	2.53	0.01	0.03	0.00
		3	0.0%	0.0%	0.0%	64.19	269.87	2.24	0.01	0.02	0.00
		5	0.0%	0.0%	0.0%	12.81	34.73	2.48	0.00	0.00	0.00
		7	0.0%	0.0%	0.0%	2.89	5.10	2.17	0.00	0.02	0.00
		10	0.0%	0.0%	0.0%	2.30	2.46	2.20	0.00	0.02	0.00
	Average		0.0%	0.0%	0.0%	131.75	252.20	49.44	0.01	0.04	0.00

Table 9. Comparison of proposed GA_{POX} performance with literature results all instances.

	G_{POX}		H_{EP-EP}		H_{MS-MS}		H_{EP-MS}		H_{MS-EP}	
Instances	$ARPD$	$TIME$	$ARPD$	$TIME$	$ARPD$	$TIME$	$ARPD$	$TIME$	$ARPD$	$TIME$
Class A	0.06%	0.28	0.97%	0.30	4.99%	0.08	0.19%	0.36	1.38%	0.32
Class B	0.11%	13.08	0.08%	27.22	9.90%	0.26	0.09%	22.59	2.15%	13.26
All	0.085%	6.310	0.525%	13.760	7.445%	0.170	0.140%	11.475	1.765%	6.790

Based on Table 9, the G_{POX} outperforms the existing heuristics since it reaches the minimum $ARPD$, which is $ARPD = 0.085\%$, for all test problems. In addition, the average computational time is satisfactory with $Time = 6.310$ s, which is ranked second after the H_{MS-MS} algorithm. Indeed, H_{MS-MS} is a polynomial heuristic. This heuristic should be discarded due to its high $ARPD$7.445%. In addition, the reduction in the $ARPD$ is $100 \times \{(0.140 - 0.085)/0.140\} = 39.29\%$ compared to the best previous heuristic, H_{EP-MS}. This provides strong evidence of the efficiency and the performance of the G_{POX} meta-heuristic. More detailed results are presented in Tables 10 and 11.

Based on Table 10, the G_{POX} outperforms the existing heuristics since it reaches the minimum average relative gap, which is $ARPD = 0.06\%$. In addition, for all combinations (n, m), G_{POX} presents the minimum $ARPD$ compared to the other heuristics. For Class B, and based on Table 10, the G_{POX} almost presents the best $ARPD$ with $GAP = 0.11\%$ versus $GAP = 0.08\%$ for the H_{EP-EP} heuristic. However, in terms of the average time, $Time = 13.08$ s for G_{POX} and $Time = 27.22$ s for H_{EP-MS}. Therefore, globally, the G_{POX} outperforms the existing heuristics. Remarkably, the average consumed time for GA_{POX} is almost linear compared to the number of jobs n. This remark could be further investigated in future research works.

Table 10. Comparison of GA_{POX} performance with existing heuristics for Class A.

m	n	$\mathbf{G_{POX}}$		$\mathbf{H_{EP-EP}}$		$\mathbf{H_{MS-MS}}$		$\mathbf{H_{EP-MS}}$		$\mathbf{H_{MS-EP}}$	
		ARPD	*TIME*	*ARPD*	*TIME*	*ARPD*	*TIME*	*ARPD*	*TIME*	*ARPD*	*TIME*
	10	0.04%	1.49	0.63%	0.01	5.82%	0.01	0.07%	0.01	0.60%	0.02
	20	0.03%	2.25	0.92%	0.01	7.45%	0.02	0.42%	0.01	1.44%	0.02
2	40	0.00%	3.74	0.20%	0.03	8.88%	0.02	0.04%	0.02	0.89%	0.03
	50	0.10%	4.43	0.49%	0.04	9.07%	0.02	0.08%	0.04	0.76%	0.06
	200	0.00%	18.72	0.34%	1.60	11.06%	0.17	0.13%	2.26	1.51%	3.28
	10	0.00%	1.52	2.22%	0.02	4.14%	0.02	0.29%	0.01	1.02%	0.01
	20	0.02%	2.31	0.89%	0.02	5.94%	0.03	0.22%	0.02	2.13%	0.03
3	40	0.02%	3.89	0.41%	0.04	5.89%	0.04	0.07%	0.03	1.93%	0.05
	50	0.00%	4.63	0.57%	0.06	6.80%	0.04	0.06%	0.05	1.85%	0.06
	200	0.00%	19.52	0.93%	1.51	8.59%	0.22	0.28%	1.76	3.69%	1.23
	10	0.00%	1.49	2.07%	0.02	1.69%	0.01	0.00%	0.01	0.82%	0.02
	20	0.17%	2.30	1.92%	0.04	2.84%	0.03	0.00%	0.03	0.80%	0.04
5	40	0.29%	3.91	0.55%	0.05	3.60%	0.06	0.17%	0.02	1.24%	0.07
	50	0.12%	4.65	0.87%	0.07	4.37%	0.05	0.20%	0.05	2.21%	0.06
	200	0.00%	19.93	1.09%	1.15	5.43%	0.29	0.43%	1.26	2.95%	0.54
	10	0.00%	1.44	0.00%	0.01	0.30%	0.02	0.00%	0.01	0.00%	0.02
	20	0.20%	2.28	1.83%	0.06	1.73%	0.05	0.01%	0.04	0.69%	0.05
8	40	0.15%	3.86	0.67%	0.17	1.90%	0.07	0.02%	0.19	0.81%	0.09
	50	0.02%	4.66	1.20%	0.09	1.74%	0.09	0.08%	0.08	0.67%	0.11
	200	0.00%	20.46	1.52%	0.98	2.61%	0.43	1.20%	1.20	1.68%	0.57
Average		0.06%	6.37	0.97%	0.30	4.99%	0.08	0.19%	0.36	1.38%	0.32

Table 11. Comparison of GA_{POX} performance with existing heuristics for Class B.

m	n	$\mathbf{G_{POX}}$		$\mathbf{H_{EP-EP}}$		$\mathbf{H_{MS-MS}}$		$\mathbf{H_{EP-MS}}$		$\mathbf{H_{MS-EP}}$	
		ARPD	*TIME*	*ARPD*	*TIME*	*ARPD*	*TIME*	*ARPD*	*TIME*	*ARPD*	*TIME*
	10	0.00%	1.36	0.00%	0.0.1	16.69%	0.02	0.00%	0.01	2.40%	0.02
	20	0.00%	2.01	0.00%	0.05	12.26%	0.03	0.00%	0.05	0.83%	0.03
2	40	0.00%	3.77	0.00%	0.05	7.23%	0.02	0.00%	0.05	0.00%	0.11
	50	0.09%	4.26	0.00%	0.17	5.84%	0.03	0.00%	0.17	0.90%	0.09
	200	0.00%	19.43	0.00%	16.78	1.13%	0.35	0.00%	20.13	0.00%	19.43
	10	0.00%	1.45	0.00%	0.01	9.39%	0.02	0.00%	0.01	2.99%	0.02
	20	0.01%	2.28	0.00%	0.01	17.70%	0.05	0.00%	0.01	5.83%	0.05
3	40	0.06%	3.85	0.00%	0.13	10.99%	0.05	0.00%	0.13	0.67%	0.19
	50	0.06%	4.33	0.00%	0.13	8.82%	0.08	0.00%	0.13	1.13%	0.59
	200	0.00%	20.93	0.01%	99.53	1.94%	0.75	0.01%	76.37	0.46%	49.31
	10	0.00%	1.47	0.00%	0.01	0.82%	0.02	0.00%	0.01	0.00%	0.02
	20	0.12%	2.36	0.00%	0.02	18.82%	0.09	0.00%	0.02	7.54%	0.08
5	40	0.22%	3.59	0.08%	25.44	15.63%	0.21	0.14%	25.42	3.16%	0.76
	50	0.19%	4.29	0.00%	12.42	13.77%	0.22	0.05%	12.43	1.68%	0.41
	200	0.03%	19.20	0.01%	91.85	3.90%	0.90	0.04%	87.51	0.45%	67.88
	10	0.00%	1.50	0.00%	0.01	0.00%	0.01	0.00%	0.01	0.00%	0.01
	20	0.00%	2.23	0.00%	0.01	0.71%	0.05	0.00%	0.01	0.00%	0.05
8	40	0.63%	3.58	1.08%	72.51	24.99%	0.43	1.16%	72.77	6.96%	0.47
	50	0.56%	4.26	0.28%	55.41	20.89%	0.36	0.28%	55.47	7.46%	0.37
	200	0.17%	18.79	0.06%	142.64	6.40%	1.41	0.19%	101.17	0.58%	125.34
Average		0.11%	6.25	0.08%	27.22	9.90%	0.26	0.09%	22.59	2.15%	13.26

6.4. Green Aspect Analysis

The objective of this study was to reduce the total consumed energy by the elimination of idle energy. This section aims to numerically evaluate the efficiency of the proposed methods to achieve this goal. In this context, for each test problem, the $\mathrm{G}_{\mathrm{POX}}$ algorithm was applied first with the no-idle constraint and then without the no-idle constraint. The two obtained respective schedules are denoted as SNI (schedule with no idle) and SI (schedule with idle). The respective makespans are denoted as C_{SNI} and C_{SI}. In addition, the consumed energy is assumed to be proportional to the duration [2]. In other terms, if T is a period of time and E is the corresponding consumed energy, then a positive real k exists, such that:

$$E = k \times T.$$

Therefore, the percent of saved idle energy, $PSIE$, is expressed as follows:

$$PSIE = \frac{I_{\mathrm{SI}}}{\sum_{i=1}^{n} p_i + I_{\mathrm{SI}}} \times 100$$

where I_{SI} is the total idle time for the schedule, SI, which is proportional to the idle energy. The expression $\sum_{i=1}^{n} p_i + I_{\mathrm{SI}}$ is proportional to the total consumed energy for the schedule SI.

In addition, PMA denotes the percent of makespan augmentation from SI to SNI. PMA is expressed as follows:

$$PMA = \frac{\mathbf{C}_{\boldsymbol{SNI}} - \mathbf{C}_{\boldsymbol{SI}}}{\mathbf{C}_{\boldsymbol{SI}}} \times 100$$

In the following table (Table 12), the average $PSIE$ ($APSIE$) and PMA ($APMA$) after running the $\mathrm{G}_{\mathrm{POX}}$ algorithm on all the test problems are presented.

Table 12. The green performance of GA_{POX} algorithm.

	APSIE	***APMA***
All	29.57%	0.12%

According to Table 12, an important reduction in the total consumed energy (29.57%) was achieved by the elimination of idle machine times. In addition, only an augmentation of 0.12% in the total cost is observed. This is strong evidence of the efficiency of the proposed procedure in saving energy and enforcing the green scheduling benefits. More detailed results are presented in Table 13.

Based on Table 12, the maximum $APSIE$ and $APMA$ were reached for ($n = 200$, $m = 2$) and ($n = 10$, $m = 2$), respectively. In addition, $APSIE$ and $APMA$ decreased when m increased. The same behavior was exhibited by $APSIE$ and $APMA$ when n increased.

Table 13. The detailed green performance of GA_{POX} algorithm.

m	***n***	***APSIE* %**	***APMA* %**
2	10	24.40%	0.91%
	20	32.14%	0.01%
	40	34.94%	0.02%
	50	34.38%	0.00%
	200	43.20%	0.00%
3	10	20.98%	0.40%
	20	28.90%	0.21%
	40	33.20%	0.00%
	50	34.51%	0.00%
	200	40.81%	0.00%

Table 13. *Cont.*

m	*n*	*APSIE* %	*APMA* %
5	10	11.89%	0.04%
	20	26.14%	0.27%
	40	31.58%	0.09%
	50	32.10%	0.00%
	200	40.62%	0.00%
8	10	5.47%	0.04%
	20	19.01%	0.30%
	40	28.40%	0.03%
	50	30.54%	0.01%
	200	38.24%	0.00%
All		29.57%	0.12%

7. Conclusions and Future Works

This study investigated a scheduling problem for *m* identical parallel machines under release dates, delivery times, and no-idle time constraints. The objective was makespan minimization. This problem is among the green scheduling problems since idle time is not permitted. Indeed, during the idle time, the machine is available without processing jobs. This is a waste of energy. By eliminating the idle time, the consumed energy is saved and minimized; this is the green aspect of this problem. In order to solve the studied problem, we propose a mixed-integer programming model (*MILP*) and a family of metaheuristics. This is due to the NP-hardness of the studied problem. The family of metaheuristics is composed of three variants of the GA and SA algorithm. The three variants of the GA are based on three different crossover operators. In addition, a lower bound (*LB*) is proposed. This lower bound was used to evaluate the produced solution. This was performed using the relative percent deviation. Extensive computational experiments were carried out to evaluate the performance and efficiency of the proposed procedures (metaheuristic MILP and *LB*). The relative percent deviation and computation time were used as performance measures. The MILP reached the optimal solution for small-sized instances. For large-sized instances, the MILP was unable to reach the optimal solution. In addition, the numerical results indicate the superiority of the proposed GA_{POX} in comparison to the MILP and the remaining proposed metaheuristics. Furthermore, it was shown that GA_{POX} outperforms the existing heuristics.

Future research works have to explore other metaheuristics and provide an efficient exact algorithm for the studied problem. The proposed procedures in this work could be useful as initial point for building such exact algorithms. In addition, more realistic constraints could be included to study problems such as the setup times, the maintenance intervention dates, or the machine unavailability constraints. Additionally, the investigation of other objective functions, such as the total tardiness minimization, could be considered.

Author Contributions: The contributions of authors are as follows: modeling, resolution, and writing, L.H.; modeling and software, A.A.; modeling and software, A.G.; software, B.B.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research at King Saud University through research group no. RG-1439-001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group no. RG-1439-001.

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

Appendix A. Tuning Plots for the Metaheuristics

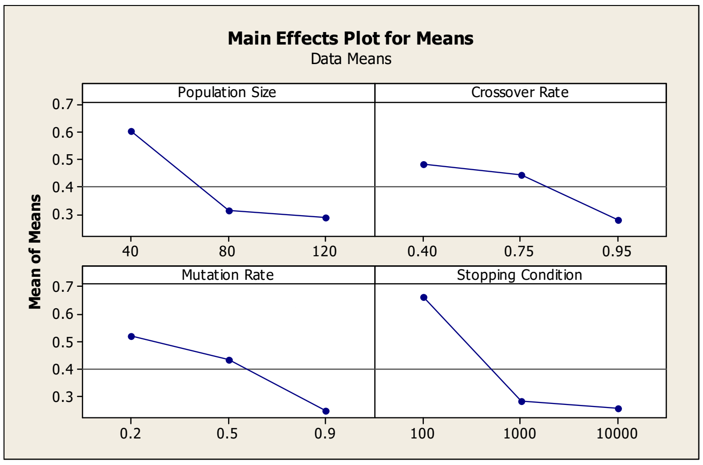


Figure A1. Parameter effect on the $ARPD$ with $n = 10$.

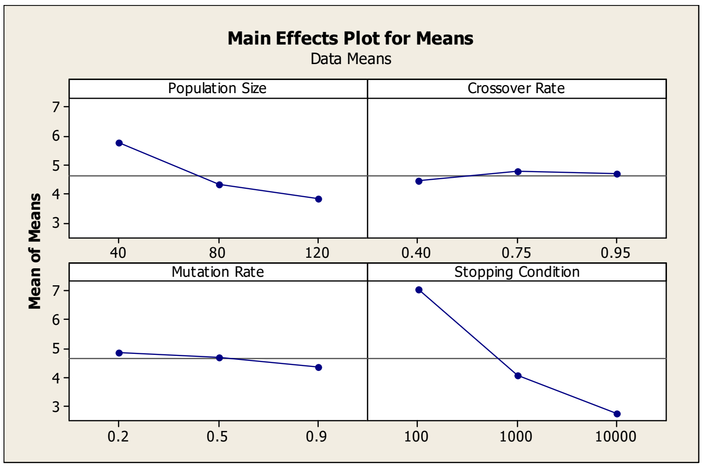


Figure A2. Parameter effect on the $ARPD$ with $n = 20$.

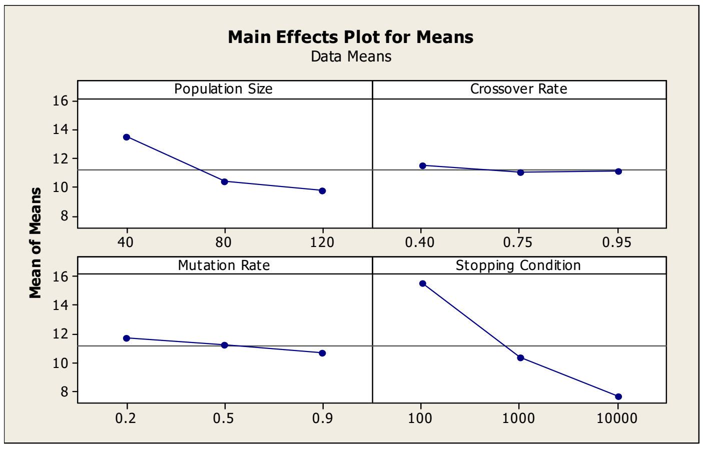


Figure A3. Parameter effect on the $ARPD$ with $n = 40$.

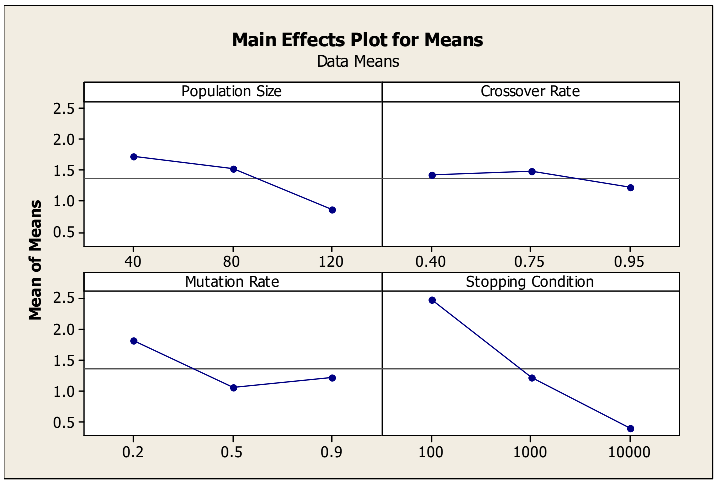


Figure A4. Parameter effect on the $ARPD$ with $n = 50$.

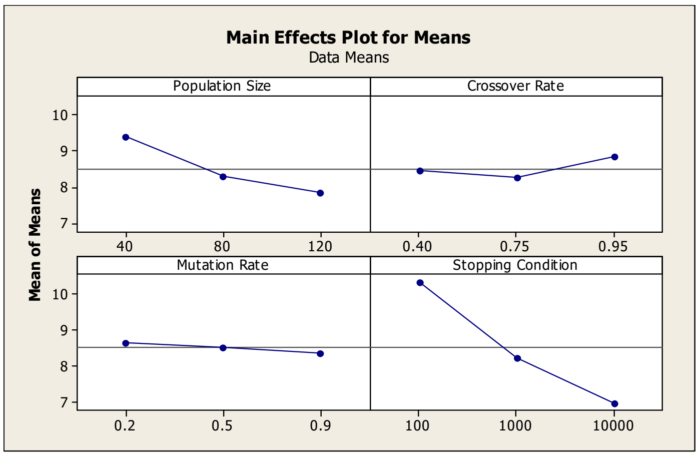


Figure A5. Parameter effect on the $ARPD$ with $n = 200$.

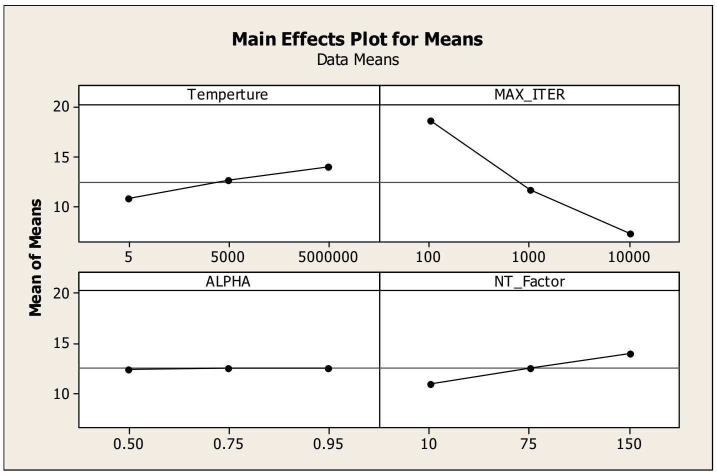


Figure A6. Parameter effect on SA results.

References

1. Cota, L.P.; Guimarães, F.G.; Ribeiro, R.G.; Meneghini, I.R.; de Oliveira, F.B.; Souza, M.J.; Siarry, P. An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem. *Swarm Evol. Comput.* **2019**, *51*, 100601. [CrossRef]
2. Gao, K.; Huang, Y.; Sadollah, A.; Wang, L. A review of energy-efficient scheduling in intelligent production systems. *Complex Intell. Syst.* **2019**, *6*, 1–13. [CrossRef]
3. Lora, A.T.; Riquelme, J.C.; Ramos, J.L.M.; Santos, J.M.R.; Expósito, A.G. Application of Evolutionary Computation Techniques to the Optimal Short-Term Scheduling of the Electrical Energy Production. In *Conference on Technology Transfer*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 656–665.
4. He, Y.; Liu, F.; Cao, H.-J.; Li, C.-B. A bi-objective model for job-shop scheduling problem to minimize both energy consumption and makespan. *J. Central South Univ. Technol.* **2005**, *12*, 167–171. [CrossRef]
5. Pinedo, M.J.L. *Scheduling: Theory, Algorithms, and Systems*; Springer Science & Business Media: New York, NY, USA, 2012.
6. Módos, I.; Šůcha, P.; Hanzálek, Z. Algorithms for robust production scheduling with energy consumption limits. *Comput. Ind. Eng.* **2017**, *112*, 391–408. [CrossRef]
7. Plitsos, S.; Repoussis, P.P.; Mourtos, I.; Tarantilis, C.D. Energy-aware decision support for production scheduling. *Decis. Support Syst.* **2017**, *93*, 88–97. [CrossRef]
8. Lei, D.; Zheng, Y.; Guo, X. A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. *Int. J. Prod. Res.* **2016**, *55*, 3126–3140. [CrossRef]
9. Mouzon, G.; Yildirim, M.B.; Twomey, J. Operational methods for minimization of energy consumption of manu-facturing equipment. *Int. J. Prod. Res.* **2007**, *45*, 4247–4271. [CrossRef]
10. Li, L.; Sun, Z.; Yao, X.; Wang, D. Optimal production scheduling for energy efficiency improvement in biofuel feedstock preprocessing considering work-in-process particle separation. *Energy* **2016**, *96*, 474–481. [CrossRef]
11. Mouzon, G.C.; Yildirim, M.B. A framework to minimise total energy consumption and total tardiness on a single machine. *Int. J. Sustain. Eng.* **2008**, *1*, 105–116. [CrossRef]
12. Benedikt, O.; Alikoç, B.; Šůcha, P.; Čelikovský, S.; Hanzálek, Z. A polynomial-time scheduling approach to minimise idle energy consumption: An application to an industrial furnace. *Comput. Oper. Res.* **2021**, *128*, 105167. [CrossRef]
13. Benedikt, O.; Šůcha, P.; Hanzalek, Z. On Idle Energy Consumption Minimization in Production: Industrial Example and Mathematical Model. *arXiv* **2005**, arXiv:2005.06270.
14. Baykasoğlu, A.; Ozsoydan, F.B. Dynamic scheduling of parallel heat treatment furnaces: A case study at a manufacturing system. *J. Manuf. Syst.* **2018**, *46*, 152–162. [CrossRef]
15. Wang, S.; Wang, X.; Yu, J.; Ma, S.; Liu, M. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *J. Clean. Prod.* **2018**, *193*, 424–440. [CrossRef]
16. Moon, I.; Jeong, Y.; Saha, S. Fuzzy Bi-Objective Production-Distribution Planning Problem under the Carbon Emission Constraint. *Sustainability* **2016**, *8*, 798. [CrossRef]
17. Jeong, Y.; Saha, S.; Chatterjee, D.; Moon, I. Direct shipping service routes with an empty container management strategy. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *118*, 123–142. [CrossRef]
18. Mokotoff, E.J.A.-P. Parallel machine scheduling problems: A survey. *Asia-Pac. J. Oper. Res.* **2001**, *18*, 193.
19. Chang, P.; Chen, S.; Lin, K. Two-phase sub population genetic algorithm for parallel machine-scheduling problem. *Expert Syst. Appl.* **2005**, *29*, 705–712. [CrossRef]
20. Chen, Z.-L.; Powell, W.B. Solving Parallel Machine Scheduling Problems by Column Generation. *Inf. J. Comput.* **1999**, *11*, 78–94. [CrossRef]
21. Cheng, R.; Gen, M. Parallel machine scheduling problems using memetic algorithms. In Proceedings of the 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929), Beijing, China, 14–17 October 1996; pp. 2665–2670.
22. Della Croce, F.; Grosso, A.; Salassa, F. Minimizing total completion time in the two-machine no-idle no-wait flow shop problem. *J. Heuristics* **2021**, *27*, 159–173. [CrossRef]
23. Akyol, D.E.; Bayhan, G.M. Minimizing makespan on identical parallel machines using neural networks. In Proceedings of the International Conference on Neural Information Processing, Hong Kong, China, 3–6 October 2006; pp. 553–562.
24. Shao, H.; Chen, H.-P.; Huang, G.Q.; Xu, R.; Cheng, B.-Y.; Wang, S.-S.; Liu, B.-W. Minimizing makespan for parallel batch processing machines with non-identical job sizes using neural nets approach. In Proceedings of the 2008 3rd IEEE Conference on Industrial Electronics and Applications, Singapore, 3–5 June 2008; pp. 1921–1924.
25. Raghavendra, B.; Murthy, A.; Rao, N. Some solution approaches to reduce the imbalance of workload in parallel machines while planning in flexible manufacturing system. *Int. J. Eng. Sci. Technol.* **2010**, *2*, 724–730.
26. Rajakumar, S.; Arunachalam, V.P.; Selladurai, V. Workflow balancing in parallel machines through genetic algorithm. *Int. J. Adv. Manuf. Technol.* **2007**, *33*, 1212–1221. [CrossRef]
27. Lee, W.-C.; Wu, C.-C.; Chen, P. A simulated annealing approach to makespan minimization on identical parallel machines. *Int. J. Adv. Manuf. Technol.* **2006**, *31*, 328–334. [CrossRef]
28. Hashemian, N.; Diallo, C.; Vizvári, B. Makespan minimization for parallel machines scheduling with multiple availability constraints. *Ann. Oper. Res.* **2012**, *213*, 173–186. [CrossRef]

29. Ouazene, Y.; Yalaoui, F.; Yalaoui, A.; Chehade, H. Theoretical Analysis of Workload Imbalance Minimization Problem on Identical Parallel Machines. In Proceedings of the Asian Conference on Intelligent Information and Database Systems, Da Nang, Vietnam, 14–16 March 2016; pp. 296–303.
30. Anghinolfi, D.; Paolucci, M.; Ronco, R. A bi-objective heuristic approach for green identical parallel machine scheduling. *Eur. J. Oper. Res.* **2021**, *289*, 416–434. [CrossRef]
31. Safarzadeh, H.; Niaki, S.T.A. Bi-objective green scheduling in uniform parallel machine environments. *J. Clean. Prod.* **2019**, *217*, 559–572. [CrossRef]
32. Saberi-Aliabad, H.; Reisi-Nafchi, M.; Moslehi, G. Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. *J. Clean. Prod.* **2020**, *249*, 119393. [CrossRef]
33. Soleimani, H.; Ghaderi, H.; Tsai, P.-W.; Zarbakhshnia, N.; Maleki, M. Scheduling of unrelated parallel machines considering sequence-related setup time, start time-dependent deterioration, position-dependent learning and power consumption minimization. *J. Clean. Prod.* **2020**, *249*, 119428. [CrossRef]
34. Li, K.; Zhang, X.; Leung, J.Y.T.; Yang, S.L. Parallel machine scheduling problems in green manufacturing industry. *J. Manuf. Syst.* **2016**, *38*, 98–106. [CrossRef]
35. Módos, I.; Šucha, P.; Hanzálek, Z. On parallel dedicated machines scheduling under energy consumption limit. *Comput. Ind. Eng.* **2021**, *159*, 107209. [CrossRef]
36. Jia, Z.-H.; Zhang, Y.-L.; Leung, J.Y.-T.; Li, K. Bi-criteria ant colony optimization algorithm for minimizing makespan and energy consumption on parallel batch machines. *Appl. Soft Comput.* **2017**, *55*, 226–237. [CrossRef]
37. Chou, Y.-L.; Yang, J.-M.; Wu, C.-H. An energy-aware scheduling algorithm under maximum power consumption constraints. *J. Manuf. Syst.* **2020**, *57*, 182–197. [CrossRef]
38. Wu, X.; Che, A. A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega* **2019**, *82*, 155–165. [CrossRef]
39. Kong, M.; Pei, J.; Liu, X.; Lai, P.-C.; Pardalos, P.M. Green manufacturing: Order acceptance and scheduling subject to the budgets of energy consumption and machine launch. *J. Clean. Prod.* **2020**, *248*, 119300. [CrossRef]
40. Abikarram, J.B.; McConky, K.; Proano, R. Energy cost minimization for unrelated parallel machine scheduling under real time and demand charge pricing. *J. Clean. Prod.* **2019**, *208*, 232–242. [CrossRef]
41. Wang, Y.; Jia, Z.-H.; Li, K. A multi-objective co-evolutionary algorithm of scheduling on parallel non-identical batch machines. *Expert Syst. Appl.* **2021**, *167*, 114145. [CrossRef]
42. Liang, P.; Yang, H.-D.; Liu, G.-S.; Guo, J.-H. An Ant Optimization Model for Unrelated Parallel Machine Scheduling with Energy Consumption and Total Tardiness. *Math. Probl. Eng.* **2015**, *2015*, 1–8. [CrossRef]
43. Antoniadis, A.; Garg, N.; Kumar, G.; Kumar, N. Parallel Machine Scheduling to Minimize Energy Consumption. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*; Society for Industrial & Applied Mathematics (SIAM): Philadelphia, PA, USA, 2020; pp. 2758–2769.
44. Brauner, N.; Kovalyov, M.Y.; Quilliot, A.; Toussaint, H. No-idle parallel-machine scheduling of unit-time jobs with a small number of distinct release dates and deadlines. *Comput. Oper. Res.* **2021**, *132*, 105315. [CrossRef]
45. Hidri, L.; Al-Samhan, A.M.; Mabkhot, M.M. Bounding Strategies for the Parallel Processors Scheduling Problem With No-Idle Time Constraint, Release Date, and Delivery Time. *IEEE Access* **2019**, *7*, 170392–170405. [CrossRef]
46. Hermès, F.; Ghédira, K. Scheduling Jobs with Releases Dates and Delivery Times on M Identical Non-idling Machines. In Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics, Madrid, Spain, 26–28 July 2017; Volume 1, pp. 82–91. [CrossRef]
47. Graham, R.; Lawler, E.; Lenstra, J.; Kan, A. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discret. Math.* **1979**, *5*, 287–326.
48. Gharbi, A.; Haouari, M. Minimizing makespan on parallel machines subject to release dates and delivery times. *J. Sched.* **2002**, *5*, 329–355. [CrossRef]
49. Soares, L.C.R.; Carvalho, M.A.M. Biased random-key genetic algorithm for scheduling identical parallel machines with tooling constraints. *Eur. J. Oper. Res.* **2020**, *285*, 955–964. [CrossRef]
50. Zhou, S.; Xie, J.; Du, N.; Pang, Y. A random-keys genetic algorithm for scheduling unrelated parallel batch processing machines with different capacities and arbitrary job sizes. *Appl. Math. Comput.* **2018**, *334*, 254–268. [CrossRef]
51. Sheremetov, L.; Martínez-Muñoz, J.; Chi-Chim, M. Two-stage genetic algorithm for parallel machines scheduling problem: Cyclic steam stimulation of high viscosity oil reservoirs. *Appl. Soft Comput.* **2018**, *64*, 317–330. [CrossRef]
52. Xiao, J.; Yang, H.; Zhang, C.; Zheng, L.; Gupta, J.N. A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times. *Comput. Oper. Res.* **2015**, *63*, 72–82. [CrossRef]
53. Liao, T.W.; Su, P. Parallel machine scheduling in fuzzy environment with hybrid ant colony optimization including a comparison of fuzzy number ranking methods in consideration of spread of fuzziness. *Appl. Soft Comput.* **2017**, *56*, 65–81. [CrossRef]
54. Lei, D.; Liu, M. An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Comput. Ind. Eng.* **2020**, *141*, 106320. [CrossRef]
55. Abdel-Basset, M.; Mohamed, R.; Abouhawwash, M.; Chakrabortty, R.; Ryan, M. A Simple and Effective Approach for Tackling the Permutation Flow Shop Scheduling Problem. *Mathematics* **2021**, *9*, 270. [CrossRef]

56. Holland, J.H. Genetic Algorithms and Adaptation. In *Adaptive Control of Ill-Defined Systems*; Springer: Boston, MA, USA, 1984; pp. 317–333.
57. Hameed, M.A.; Jamsheela, O.; Robert, B.S. Relative performance of Roulette wheel GA and Rank GA is dependent on chromosome parity. *Mater. Today Proc.* **2021**, in press. [CrossRef]
58. Koohestani, B. A crossover operator for improving the efficiency of permutation-based genetic algorithms. *Expert Syst. Appl.* **2020**, *151*, 113381. [CrossRef]
59. Nitisiri, K.; Gen, M.; Ohwada, H. A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling. *Comput. Ind. Eng.* **2019**, *130*, 381–394. [CrossRef]
60. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]