processes

MDPI

Article

Scheduling Large-Size Identical Parallel Machines with Single Server Using a Novel Heuristic-Guided Genetic Algorithm (DAS/GA) Approach

Mohammad Abu-Shams [1,*], Saleem Ramadan [2], Sameer Al-Dahidi [3] and Abdallah Abdallah [1]

[1] Industrial Engineering Department, School of Applied Technical Sciences, German Jordanian University, Amman 11180, Jordan
[2] Mechanical, Industrial & Manufacturing Engineering, Youngstown State University, Youngstown, OH 44555, USA
[3] Mechanical and Maintenance Engineering Department, School of Applied Technical Sciences, German Jordanian University, Amman 11180, Jordan
* Correspondence: mohammad.abushams@gju.edu.jo; Tel.: +962-4294444 (ext. 4508)

Abstract: Parallel Machine Scheduling (PMS) is a well-known problem in modern manufacturing. It is an optimization problem aiming to schedule *n* jobs using *m* machines while fulfilling certain practical requirements, such as total tardiness. Traditional approaches, e.g., mix integer programming and Genetic Algorithm (GA), usually fail, particularly in large-size PMS problems, due to computational time and/or memory burden and the large searching space required, respectively. This work aims to overcome such challenges by proposing a heuristic-based GA (DAS/GA). Specifically, a large-scale PMS problem with *n* independent jobs and *m* identical machines with a single server is studied. Individual heuristic algorithms (DAS) and GA are used as benchmarks to verify the performance of the proposed combined DAS/GA on 18 benchmark problems established to cover small, medium, and large PMS problems concerning standard performance metrics from the literature and a new metric proposed in this work (standardized overall total tardiness). Computational experiments showed that the heuristic part (DAS-*h*) of the proposed algorithm significantly enhanced the performance of the GA for large-size problems. The results indicated that the proposed algorithm should only be used for large-scale PMS problems because DAS-*h* trapped GA in a region of local optima, limiting its capabilities in small- and mainly medium-sized problems.

Keywords: scheduling; optimization; heuristic; genetic algorithm; identical parallel machines; apparent tardiness cost rule

Citation: Abu-Shams, M.; Ramadan, S.; Al-Dahidi, S.; Abdallah, A. Scheduling Large-Size Identical Parallel Machines with Single Server Using a Novel Heuristic-Guided Genetic Algorithm (DAS/GA) Approach. *Processes* **2022**, *10*, 2071. https://doi.org/10.3390/pr10102071

Academic Editors: Danyu Bai, Xin Chen, Dehua Xu and Jedrzej Musial

Received: 13 September 2022
Accepted: 11 October 2022
Published: 13 October 2022

1. Introduction

In modern manufacturing, the Parallel Machine Scheduling (PMS) problem amounts to scheduling several jobs using various identical machines while fulfilling specific practical requirements, such as minimum total tardiness while executing the jobs [1–5]. Thus, the PMS problem can be formulated as an NP-hard optimization problem that requires sophisticated optimization techniques for scheduling the jobs using the available machines while satisfying some practical constraints [6–10].

Many algorithms have been proposed in the literature to deal with the PMS problem. According to refs. [11–15], the algorithms used in PMS can be globally divided into two main groups: construction and improvement or interchange algorithms. The construction algorithms choose one job at a time and fix it in the available position using dispatching rules. One of the well-known construction rules is the Apparent Tardiness Cost (ATC) rule [16]. Several scholars have used modified versions of ATC to minimize the total tardiness in PMS [17–23].

On the other hand, the improvement or interchange algorithms work on an initial solution and use local interchanges to improve the solution. Several scholars have used

meta-heuristics as improvement algorithms, such as Genetic Algorithms (GAs). GAs have been heavily used as improvement algorithms for PMS problems [24–40]. Simulated Annealing (SA) [41,42], Ant Colony (AC) [43–45], Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [7,46–48], NSGA-III [1,49,50], and Non-dominated Ranking GA (NRGA) [51] have also been used to optimize PMS as improving algorithms.

For example, Wang et al. [31] proposed modified versions of two meta-heuristics algorithms (GA and SA) to obtain an approximate solution to large-scale PMS problems with unrelated parallel machines. Sharma et al. [52] proposed and evaluated the effectiveness of a multi-step crossover GA on randomly generated PMS problems of different sizes using identical parallel machines. Laha et al. [42] proposed a SA meta-heuristics algorithm for minimizing makespan in identical PMS problems. Jia et al. [45] proposed a fuzzy AC optimization algorithm to obtain better solutions within a convenient time for fuzzy scheduling problems (i.e., jobs characterized by fuzzy processing time) on parallel batch machines with different capacities. Farmand et al. [8] investigated the effectiveness of two meta-heuristic algorithms (Multi-Objective Particle Swarm Optimization (MOPSO) and NSGA-II) on small, medium, and large-scale PMS problems with identical parallel machines.

In this paper, we consider a large-size identical parallel machine scheduling problem P_m with n independent jobs and m identical machines with a single server. Traditional computational algorithms, such as Mix Integer Programming (MIP) and GA, usually fail or perform badly in such large-size problems due to computational time limitations and/or memory limitations and the large search space required, respectively. A meta-heuristic algorithm DAS/GA is developed in this paper to overcome these limitations while further boosting the performance of the GA. The problem can be described as follows: given a sequence of n independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ and a set of m identical machines $M = \{M_1, M_2, \ldots, M_m\}$, the objective is to sequence the jobs on the machines to minimize the total tardiness $\sum_i T_i$ of the schedule. Job J_i is associated with a processing time PT_i and a due date DD_i with an independent setup time ST_{si}, which is included in PT_i. One server S_1 exists for dispatching the jobs to the available machines according to the dispatching rules in the algorithm. The standard notation for this problem can be denoted as $P_m, S1|ST_{si}|\sum_i T_i$ according to ref. [53], which is known to be an NP-hard problem [54]. Individual heuristic algorithms (DAS) and GA are used as benchmarks to assess the effectiveness of the proposed combined DAS/GA algorithm on 18 benchmark problems selected to represent small, medium, and large PMS problems.

The assumptions made in this article are as follows: the jobs are independent; each job has a single operation; the processing times include the corresponding setup times, and the setup times are independent of the sequence; machines are identical and have 100% availability and utilization while jobs are waiting; no preemption, no cancelation, and no priority for jobs are allowed; all of the jobs are available at time zero, and the problem is static and deterministic.

The rest of the article is organized as follows: Section 2 presents the framework of the proposed heuristic-guided GA; Section 3 discusses the performance measures used to evaluate the effectiveness of the proposed algorithm to the benchmarks; Section 4 explains the data generation method for the experimentation problems setup; Section 5 discusses the results of the application; and finally, Section 6 concludes the article and highlights future work.

2. The Framework of the Proposed Heuristic-Guided Genetic Algorithm (DAS/GA)

The meta-heuristic algorithm DAS/GA proposed in this article consists of heuristic DAS-*h*, followed by GA. DAS-*h* itself consists of 3 heuristics working sequentially: Due Date Tightness heuristic (DDT-*h*), Apparent Tardiness Cost heuristics (ATC-*h*), and Swap heuristic (S-*h*). Figure 1 shows the flowchart of the proposed DAS/GA.

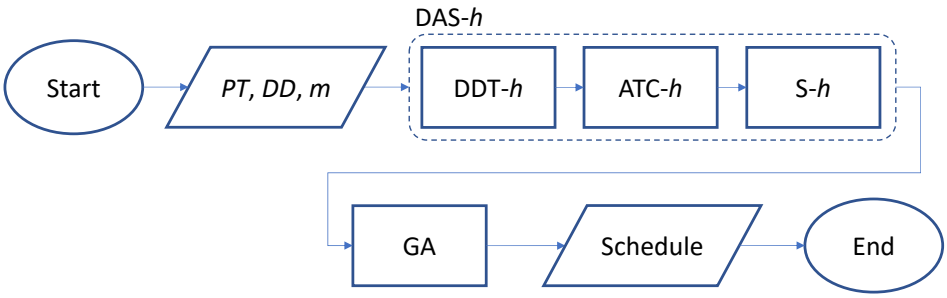

Figure 1. The flowchart of the proposed DAS/GA.

DAS/GA starts with the construction heuristic (DDT-*h*), which assigns jobs to different machines based on their due dates' tightness index values. Then, the produced schedule is fed into the modified version of the ATC heuristic (ATC-*h*) to further improve the schedule. The produced schedule is fed into a third improvement heuristic (S-*h*), which will fine-adjust the schedule. The output schedule of S-*h* is used as one chromosome (i.e., schedule) to seed the initial population in the GA. The heuristics and the GA comprising DAS/GA meta-heuristic are discussed in detail in the following sections.

2.1. Due Date Tightness Heuristic (DDT-h)

In DDT-*h*, jobs are assigned to different machines based on their Due Date Tightness index I_i^{DD} values. This index is calculated by Equation (1), in which the DD_i and PT_i are the due date and the processing time for job i, respectively. DDT-*h* sorts the jobs in an ascending order based on their I_i^{DD} values such that a smaller I_i^{DD} value indicates a higher priority job.

$$I_i^{DD} = \frac{DD_i - PT_i}{PT_i} \quad (1)$$

To illustrate how the DDT-*h* works, consider 2 machines and 10 jobs along with their processing times and due dates, as shown in Table 1. Their I_i^{DD} values were calculated according to Equation (1) and are recorded in Table 1.

Table 1. Data and I_i^{DD} values for the illustrative example.

Job *i*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PT_i	45	894	275	840	357	50	653	912	257	224
DD_i	65	906	321	1528	378	69	692	1490	342	373
I_i^{DD}	0.444	0.013	0.167	0.819	0.059	0.380	0.060	0.634	0.331	0.665

The 10 jobs are sorted in ascending order based on their I_i^{DD} values, as reported in Table 2.

Table 2. Sorted jobs based on I_i^{DD} values.

Job *i*	**2**	**5**	**7**	**3**	**9**	**6**	**1**	**8**	**10**	**4**
I_i^{DD}	0.013	0.059	0.060	0.167	0.331	0.380	0.444	0.634	0.665	0.819

After jobs were sorted, the job with the smallest I_i^{DD} value is dispatched as the first job on the first machine, the job with the second-smallest I_i^{DD} value is dispatched as the first job on the second machine, the job with third-smallest I_i^{DD} value is dispatched as the second job on the first machine, and so on. The resulting schedule is summarized in Table 3.

Table 3. DDT-*h* schedule.

Position	1	2	3	4	5
Machine 1	2	7	9	1	10
Machine 2	5	3	6	8	4

2.2. Apparent Tardiness Cost Heuristic (ATC-h)

The Apparent Tardiness Cost heuristic (ATC-*h*) is an iterative heuristic that schedules jobs one at a time from a set of remaining available jobs. The heuristic contains a priority index $I_i(t)$ that changes dynamically with time t. Variations of ATC-*h* were explained, among other references, in refs. [17,20]. We highlight the main aspects of ATC-*h* in this article for convenience and modify it to suit the $P_m,\ S1|ST_{si}|\sum_i T_i$ problem at hand. The $I_i(t)$ is calculated by multiplying two terms, which are the Weighted Shortest Processing Time ($WSPT$) and the Least Slack (LS), as in Equation (2):

$$I_i(t) = WSPT \times \text{LS} \tag{2}$$

The Weighted Shortest Processing Time ($WSPT$) is given by Equation (3):

$$WSPT = \frac{1}{PT_i}, \tag{3}$$

and the LS is given by Equation (4):

$$\text{LS} = e^{-\left(\frac{max(DD_i - PT_i - CT_k\ ,\ \ 0)}{\zeta \times \mu_{PT}}\right)}, \tag{4}$$

where CT_k is the cumulative time for predecessor job k, μ_{PT} is the mean of the processing times for the remaining jobs, and ζ is a look-ahead parameter, which is determined through experimentation. According to ref. [17], ζ can be calculated as in Equation (5):

$$\zeta = 1.2 \times \ln\left(\frac{n}{m}\right) - \frac{\left(\underset{i}{maxDD_i} - \underset{i}{minDD_i}\right)m}{n\mu_{PT}} \tag{5}$$

Substituting Equations (3) and (4) in Equation (2) gives the final equation for $I_j(t)$ as in Equation (6):

$$I_i(t) = \frac{1}{PT_i} \times e^{-\left(\frac{max(DD_i - PT_i - CT_k\ ,\ \ 0)}{\zeta \times \mu_{PT}}\right)}, \tag{6}$$

where ζ is given by Equation (5). It should be noted that the processing time for a job includes its setup time, and the setup time for the job is independent of the sequence that includes the job, as stated in assumption 3 in this article. Hence, the equations reported in ref. [17], which assume a dependent setup time to derive $I_i(t)$, should be modified from their original form in ref. [17] to Equations (4)–(6).

As seen in Equation (3), $WSPT$ is the inverse of the processing times. This means that jobs with lower processing times have higher priority (providing everything else remains the same in LS). Moreover, Equation (4) shows that LS has two possible values: value 1 when the slack value for the job is negative, which indicates that the job is tardy, or any value between 0 and 1 when the slack value is not negative, which indicates that the job is not tardy. This means that a tardy job has the highest possible LS value and consequently has the highest possible $I_i(t)$ value among the jobs with equal processing times.

Every time a machine is available, the priority indices values of all the remaining jobs are re-calculated, and the job with the highest $I_i(t)$ value is chosen to be processed next. The decision is dynamic, as the $I_i(t)$ value of the job changes over time because it depends on CT_k, the cumulative time for its predecessor.

The ATC-*h* needs an initial schedule to work on. This matter makes the final schedule for ATC-*h* dependent on the initial schedule provided. Table 4 shows a possible initial schedule for ATC-*h*.

Table 4. Initial schedule for ATC-*h*.

Position	**1**	**2**	**3**	**4**	**5**
Machine 1	4	5	6	1	2
Machine 2	10	9	7	8	3

Table 5 shows the $I_i(t)$ values generated for the data in Table 1 based on the initial schedule provided in Table 4.

Table 5. $I_i(t)$ values for data in Table 1 and schedule in Table 4.

Machine 1						
Job Iteration	**4**	**5**	**6**	**1**	**2**	**Remarks**
1	0.0003	0.0027	0.0193	0.0214	0.0011	Job 1 first
2	0.0004	0.0028	0.0200	−1	0.0011	Job 6 second
3	0.0004	0.0028	−1	−1	0.0011	Job 5 third
4	0.0004	−1	−1	−1	0.0011	Job 2 fourth
5	0.0004	−1	−1	−1	−1	Job 4 fifth
Machine 2						
Job Iteration	**10**	**9**	**7**	**8**	**3**	**Remarks**
1	0.0036	0.0034	0.0014	0.0005	0.0034	Job 10 first
2	−1	0.0039	0.0015	0.0006	0.0036	Job 9 second
3	−1	−1	0.0015	0.0009	0.0036	Job 3 third
4	−1	−1	0.0015	0.0011	−1	Job 7 fourth
5	−1	−1	−1	0.001	−1	Job 8 fifth

Table 6 shows the schedule produced using ATC-*h*.

Table 6. ATC-*h* schedule.

Position	**1**	**2**	**3**	**4**	**5**
Machine 1	1	6	5	2	4
Machine 2	10	9	3	7	8

Figure 2 shows the standardized values of $I_i(t)$ and I_i^{DD} for the 10 jobs described in Table 1. In ATC-*h*, as $DD_i - PT_i$ value, the Due Date Tightness (DDT), increases, the $I_i(t)$ value decreases, and consequently the priority of the job decreases, while in DDT-*h*, as the DDT value increases, the I_i^{DD} value increases but the priority of the job decreases.

The Figure shows that the two heuristics have the same basic behavior for jobs with small values of DDT like jobs 2, 3 and 7 but have fundamentally different behavior for jobs with large values like jobs 4 and 8. This difference in the behavior is because the ATC-*h* takes into account the cumulative processing time for the predecessor CT_k in calculating LS while DDT-*h* does not. The effect of the CT_k on the $I_i(t)$ values in ATC-*h* is evident in jobs 2 and 8, where both jobs have very close $I_i(t)$ values even though they have a big difference in their DDT values.

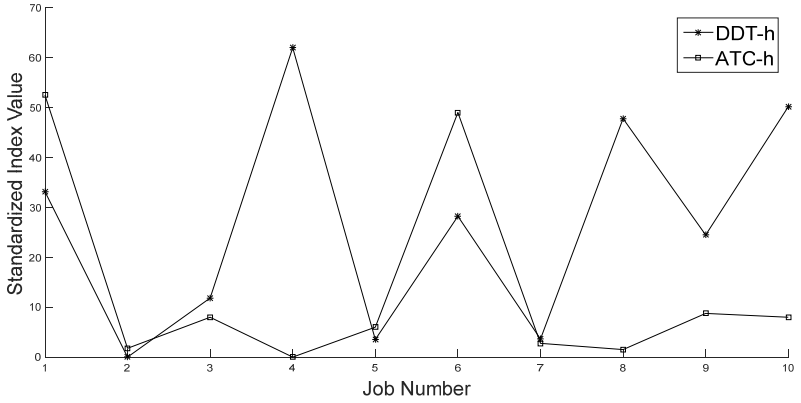

Figure 2. The standardized values of $I_i(t)$ and I_i^{DD} for data in Table 1.

2.3. Swap Heuristic (S-h)

The aim of the Swap Heuristic (S-*h*) is to fine-adjust the schedule by removing one job (with positive tardiness) at a time from a tardy machine and assigning it to the machine with the lowest total tardiness. This heuristic is a fine-adjusting heuristic, so it is meant to be applied after ATC-*h*, which is a coarse-adjusting heuristic. The aim of this heuristic is to level the load on the machines. S-*h* is similar to ATC-*h* in terms of the need for an initial schedule and in terms of using the cumulative times in their calculations.

The pseudo-code for this heuristic is as follows (Algorithm 1):

Algorithm 1: The pseudo-code of the Swap Heuristic (S-*h*)

Input schedule from ATC-*h*
Calculate the tardiness T_j for each machine in machine set M
Determine the machine M_L with lowest T_j
Determine the set that contains the rest of the other machines M_o such that $M_{o_j} \in \{x \in M | x \notin M_L\}$
Determine the set of jobs J_L on M_L
Calculate $C_{t_l} = \sum_{i \in J_l} PT_i$
Do WHILE the overall total tardiness TT is improving or termination condition is reached
    FOR all machines indexed j in M_o do the following
        FOR all jobs indexed i in M_{o_j} do the following
            Calculate the cumulative processing time $C_{t_{j_i}}$ for job i on M_{o_j}
            IF $C_{t_l} + PT_i < C_{t_{j_i}}$ do the following
                Remove job i from M_{o_j} and assign it to the end of the schedule for M_L
                Update T_j, M_L, J_L,M_o, and C_{t_l}
                Go to DO WHILE loop
            END IF
        END FOR indexed i
    END FOR indexed j
END DO WHILE
Report the new Schedule

The heuristic chooses the job to be moved according to the Ineq. 7 (Equation (7)).

$$C_{t_l} + PT_i < C_{t_{j_i}} \tag{7}$$

Ineq. 7 states that moving a tardy job from any machine in M_o to the end of the schedule for M_L always results in an improvement in the overall tardiness of the schedule. The left-hand side of the inequality is a quasi-representation of the new tardiness value for the removed job i in the new machine, while the right-hand side is a quasi-representation of the tardiness value for the removed job i in its original machine. If the left-hand side is less than the right-hand side of the inequality, then removing job i from the current machine M_{o_j} and assigning it to M_L guarantees enhancement of the overall total tardiness of the schedule by the difference between the left-hand side and the right-hand side values.

To illustrate the effect of this heuristic, consider the data in Table 7 and the corresponding proposed schedule in Table 8.

Table 7. Process times and due dates for Table 8.

Job *i*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PT_i	100	1150	275	840	357	50	750	912	450	224
DD_i	165	1200	321	1528	378	69	692	1490	342	373

Table 8. The proposed schedule.

Position	**1**	**2**	**3**	**4**	**5**
Machine 1	2	7	9	1	10
Machine 2	5	3	6	8	4

The total tardiness for machine 1 is 7802 and for machine 2 is 1934; the overall total tardiness for the schedule is 9736. The cumulative processing time for machine 2, which is M_L in this case, is 2434; hence, feeding the data of job 10 in Ineq. 7 gives a correct inequality, as shown below:

$$(2434 + 224 - 373) < (2674 - 373) \rightarrow 2285 < 2301$$

Thus, removing job 10 from machine 1 and assigning it at the end of the schedule for machine 2 should improve the overall total tardiness for the schedule by 2301 − 2285 = 16. In fact, removing job 10 from machine 1 and assigning it at the end of the schedule for machine 2 gives an overall total tardiness of 9720, which is 16 units less than the total tardiness of the original schedule, with total tardiness for machine 1 of 5501 and total tardiness for machine 2 of 4219. It should be noticed here how S-*h* levels the load on the machines and enhances the overall total tardiness of the schedule.

2.4. Genetic Algorithm (GA)

The GA has been used intensively in PMS with the aim of minimizing the overall total tardiness of the schedule [25,33–35]. GA has two kinds of operations: the genetic operation and the evolution operation. Crossover and mutation in GA belong to the genetic operation, while the selection mechanism belongs to the evolution operation [55]. The crossover operator in GA aims to roughly search the solution space while the mutation operator aims to finely search the solution space by exploiting the promising areas found by the crossover operator [56].

Chromosome representation, mutation, and selection strategies should be tailored to fit the problem at hand. These strategies are explained in the following sections.

2.4.1. Chromosome Representation

In this proposed GA, the chromosome consists of m rows, one row for each machine, and $n - (m - 1)$ columns (positions) for each row, to ensure that each machine has at least one job on it. In this representation, gene $\xi_{ij} = k$ means that the i^{th} position on the j^{th} machine is occupied by job k. In this representation, each gene carries three pieces of information: the value of the gene itself represents the job, while the position of the gene, determined by i and j indices, represents the machine and the position on that machine for the job. This representation ensures the feasibility of the chromosomes and offspring and thus avoids the need for any repair actions.

Consider the data used in Table 1. Table 9 shows one possible chromosome for this data. The phenotype can be easily retrieved from the genotype in this representation, as gene $\xi_{22} = 5$ means that job 5 is the second job to be processed on machine 2, while $\xi_{17} = 0$ means that there are no jobs processed in position 7 or beyond for machine 1.

Table 9. One possible chromosome for data in Table 1.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Machine 1	2	7	6	3	1	9	0	0	0
Machine 2	10	5	8	4	0	0	0	0	0

2.4.2. Fitness Function

In this GA, the fitness function used is the overall total tardiness of the schedule, as expressed in Equation (8):

$$TT = \sum_{j=1}^{m} \sum_{\substack{d_{\xi_{ij}} < (S_{\xi_{ij}} + PT_{\xi_{ij}}) \\ i=1:n-(m-1)}} \left(S_{\xi_{ij}} + PT_{\xi_{ij}} - DD_{\xi_{ij}} \right), \tag{8}$$

where $S_{\xi_{ij}}$, $PT_{\xi_{ij}}$, and $DD_{\xi_{ij}}$ are the start time, processing time, and due date for gene ξ_{ij}.

The fitness function value for the chromosome in Table 9 is as follows:

$$\sum_{j=1}^{2} \sum_{\substack{d_{\xi_{ij}} < (S_{\xi_{ij}} + PT_{\xi_{ij}}) \\ i=1:5}} \left(S_{\xi_{ij}} + PT_{\xi_{ij}} - DD_{\xi_{ij}} \right) = (7618) + (1011) = 8629.$$

2.4.3. Mutation

This GA is a crossover-free GA in which only mutation is used to produce the offspring from a single parent. Four different mutation types were used to mutate the selected chromosome. The Two Genes Exchange mutation (TGEm) affects only two jobs on the same machine, as it chooses two random jobs from a randomly selected machine and switches their positions. The Number of Jobs mutation (NoJm) changes the number of jobs on two randomly selected machines as it swaps randomly selected jobs from one randomly selected machine to another randomly selected one, provided that the minimum of one job per machine constraint is conserved. NoJm imitates the S-*h*, but it differs from S-*h* in two aspects. First, S-*h* swaps one job each time it is applied while NoJm may swap more than one job each time it is applied. Second, unlike S-*h*, NoJm does not guarantee that the change is beneficial. The Flip Ends mutation (FEm) flips the ends of the schedule for a randomly selected machine such that the first job becomes the last job and the last job becomes the first job on that machine. Flip Middle mutation (FMm), flips the sequence of the jobs between two randomly selected positions near the middle of the machine's schedule.

TGEm plays the role of the traditional mutation in the GA, in which it generates a limited disturbance in the chromosome; hence, it performs fine search. The other three types of mutations play the role of crossover in the GA, as they introduce high disturbances in the chromosome; hence, they play the role of coarse search. The offspring produced by these four types of mutations are always feasible offspring thanks to the chromosome representation discussed earlier. Consequently, there is no need for any repair actions on the offspring. Moreover, in this GA, a 25% mutation rate is adopted. This means that the number of offspring generated by mutation is the same as the population size, as each chromosome selected for the mutation will produce 4 offspring.

2.4.4. Selection

An elitist selection strategy is adopted in this GA. Under this strategy, all the parents and the offspring form a pool in which they have to compete for their survival. Those who have better fitness values will be selected as the parents of the next generation.

2.5. Mathematical Model

Binary programming for $P_m,\ S1|ST_{si}|\sum_i T_i$ is formulated in studies such as refs. [57,58]. The objective function of this model is

$$TT = \sum_{k=1}^{m}\sum_{j=1}^{p} T_{jk} \tag{9}$$

The constraints are

$$\sum_{j=1}^{p}\sum_{k=1}^{m} X_{ijk} = 1 \qquad ,\forall\ i = 1, \ldots,\ n \tag{10}$$

$$\sum_{i=1}^{job} X_{ijk} \leq 1 \qquad ,\forall\ \ j = 1, \ldots,\ p\ and\ k = 1, \ldots,\ m \tag{11}$$

$$C_{jk}\ =\ C_{[j-1]k}\ + \sum_{i=1}^{n} PT_i * X_{ijk}\ \forall\ j = 1,\ \ldots,\ p,\quad k = 1,\ \ldots,\ m.\ C_{0k} = 0 \tag{12}$$

$$C_{jk}\ - \sum_{i=1}^{n} DD_i * X_{ijk} - T_{jk} \leq\ 0 \quad \forall\ j = 1,\ \ldots,\ p,\quad \forall k = 1,\ \ldots,\ m \tag{13}$$

$$T_{jk} \geq 0 \tag{14}$$

$$X_{ijk\ Binary} \tag{15}$$

In this mathematical model, indices i, j, and k are indices for job, position, and machine, respectively, and p is the number of positions on the machine. Moreover, $X_{ijk} = 1$ if job i is processed in position j at machine k and is "0" otherwise, C_{jk} is the completion time for position j at machine k, T_{jk} is a positive value that represents the tardiness value for position j at machine k.

Equation (9) calculates the overall total tardiness of the schedule by summing the individual tardiness values for the positions on the machines. Equation (10) guarantees that each job is assigned only once, while Equation (11) guarantees that each position on each machine, if occupied, will be occupied only once. Equation (12) calculates the cumulative processing time for the job, and Equations (13) and (14) together dictate that if the position on the machine is occupied by a tardy job, then the tardiness of the position equals the tardiness of that job; otherwise, the tardiness of the position and consequently the tardiness of the job is zero.

3. Performance Measures

The performance of DAS-h, DAS/GA, and GA was measured in this article using two performance measures suggested by ref. [17], (Relative Error (RE) and Average Relative Improvement (ARI)), and a third performance measure that is suggested in this article: the standardized overall total tardiness $StdrdTT$.

RE is the difference between the tardiness of the method used TT_h and the tardiness of the optimal schedule TT_{op} found by the binary programming model discussed in Section 2.5 using CPLEX software relative to TT_{op}. Mathematically RE is given by Equation (16). It should be noted that this measure can only be used when $TT_{op} > 0$. If TT_{op} could not be found due the memory limitations or if $TT_{op} = 0$, TT_{op} will be substituted by TT_{Best}, which is the best TT found among the different methods used to solve the problem.

$$RE = \frac{TT_h - TT_{op}}{TT_{op}} \tag{16}$$

ARI is used only with GA and DAS/GA to measure their tardiness $TT_{GA|DAS/GA}$ with respect to the tardiness of DAS-h, which is $TT_{DAS\text{-}h}$. Mathematically, ARI is given by Equation (17):

$$ARI = \frac{TT_{GA|DAS/GA}}{TT_{DAS\text{-}h}} \tag{17}$$

$StdrdTT$ is the relative deviation between the overall total tardiness of the method and the minimum overall total tardiness among the methods used, divided by the overall total tardiness among the methods used. Mathematically, $StdrdTT$ is given by Equation (18):

$$StdrdTT = \frac{TT_H - min(TT_{DAS\text{-}h}, TT_{DAS/GA}, TT_{GA})}{min(TT_{DAS\text{-}h}, TT_{DAS/GA}, TT_{GA})}, \quad \forall\, H \in \{DAS\text{-}h,\ DAS/GA,\ GA\} \tag{18}$$

It should be noted that RE and $StdrdTT$ is the same for cases where $TT_{op} = 0$.

4. Data Generation for Experiments

In this section, 18 benchmark problems are considered for experimentation. The problems' instances were generated such that they cover small, medium, and large problems to study the performance of the different methods, ATC-h, DAS, DAS/GA, and GA, under these cases. The average of 20 replicates was considered in calculating the performance of GA, DAS/GA, and ATC-h, as they demand a random start-up schedule. The running time for DAS/GA and GA is only 1 min per replicate. The instances for the experimentation problems were generated according to Fisher's standard method as discussed in refs. [57,59]. The method starts by generating n integer processing times $PT_i s$ from a uniform distribution such that $PT_i \sim U[1, 100]$. The corresponding due dates $DD_i s$ are generated from another uniform distribution such that $DD_i \sim \left[P\left(1 - \tau - \frac{R}{2}\right)/m,\ P\left(1 - \tau + \frac{R}{2}\right)/m\right]$, where $P = \sum_{i=1}^{n} PT_i$, $\tau \in \{0.2,\ 0.4,\ 0.6,\ 0.8,\ 1\}$ and $R \in \{0.2,\ 0.4,\ 0.6,\ 0.8,\ 1\}$. The template used for the instances is $J\_M\_\tau\_R$. For example, 2000_10_04_04 means scheduling 2000 jobs on 10 identical machines such that due dates are generated using $\tau = 0.4$ and $R = 0.4$.

5. Results and Experiments

Table 10 shows the performance measures for 18 problems generated according to Fisher's standard method as discussed earlier. It should be noted that CPLEX software did not find the optimal solution, Opt. TT, for problems beyond problem 11 due to memory limitations. This shows the importance of this work and other related works in solving big NP-hard PMS problems where binary programming fails due to technical issues.

Figure 3a shows a comparison between the performances of the different methods using the RE measure. Figure 3b shows that the performance of GA slightly outperformed the performance of DAS/GA for small problems and significantly for medium problems. On the other hand, for large problems, the performance of DAS/GA significantly outperformed the performance of GA. This means that DAS-h helped GA improve its performance significantly in large problems but deteriorated its performance for medium problems. Figure 3c shows that the trend between the performances of GA and DAS-h is the same as the trend captured in Figure 4b between GA and DAS/GA. The performance of GA slightly outperformed the performance of DAS-h for small problems and significantly for medium problems, while for large problems the performance of DAS-h significantly outperformed the performance of GA. From Figure 3b,c, one can see that the behavior of DAS-h masks the behavior of DAS/GA for large problems, as DAS/GA has the same behavior of DAS-h relative to GA in this range of problem sizes. The comparison between the performance of DAS-h and DAS/GA shown in Figure 3d supports this argument. The Figure shows that DAS/GA slightly outperformed DAS-h for small problems and significantly for medium problems; the methods had a negligible difference in their performance for large problems.

Table 10. Results for the 18 problems used in the experimentation.

#	Problem	Opt.	DAS		DAS/GA			GA		
		TT	*TT*	*RE*	*TT*	*RE*	*ARI*	*TT*	*RE*	*ARI*
1	20_02_02_02	147	162	0.1020	151.8	0.0327	0.9370	151.7	0.0320	0.9364
2	20_05_02_02	149	195	0.3087	161.7	0.0852	0.8292	160.1	0.0745	0.8210
3	20_10_10_10	195	246	0.2615	226.0	0.1590	0.9187	218.1	0.1185	0.8866
4	30_02_02_02	83	188	1.2651	83.0	0.0000	0.4415	83.0	0.0000	0.4415
5	30_10_02_02	101	787	6.7921	171.4	0.6970	0.2178	170.7	0.6901	0.2169
6	40_02_02_02	13	26	1.0000	13.0	0.0000	0.5000	13.0	0.0000	0.5000
7	40_05_02_02	68	166	1.4412	102.8	0.5118	0.6193	85.6	0.2588	0.5157
8	40_10_02_02	0	0	NA	0.0	NA	NA	0.0	NA	NA
9	50_05_02_02	49	188	2.8367	104	1.1224	0.5532	102.2	1.0857	0.5436
10	100_02_02_02	25	141	4.6400	25	0.0000	0.1773	25.0	0.0000	0.1773
11	100_5_02_02	86	240	1.7907	140.7	0.6360	0.5863	104.4	0.2140	0.4350
12	100_10_02_02	NA	195	0.0285	189.6	0.0000	0.9723	190.0	0.0021	0.9744
13	300_15_06_06	NA	28164	0.0084	27929	0.0000	0.9917	41182.0	0.4745	1.4622
14	500_10_05_05	NA	50428	0.0041	50220	0.0000	0.9959	74571.0	0.4849	1.4788
15	750_15_04_04	NA	47727	0.0004	47707	0.0000	0.9996	73344.0	0.5374	1.5367
16	1000_10_06_06	NA	355500	0.0003	355396	0.0000	0.9997	590010.1	0.6601	1.6597
17	1500_15_04_04	NA	168559	0.0011	168378.7	0.0000	0.9989	274604.3	0.6309	1.6291
18	2000_10_04_04	NA	399815	0.0004	399673	0.0000	0.9996	649413.5	0.6249	1.6243

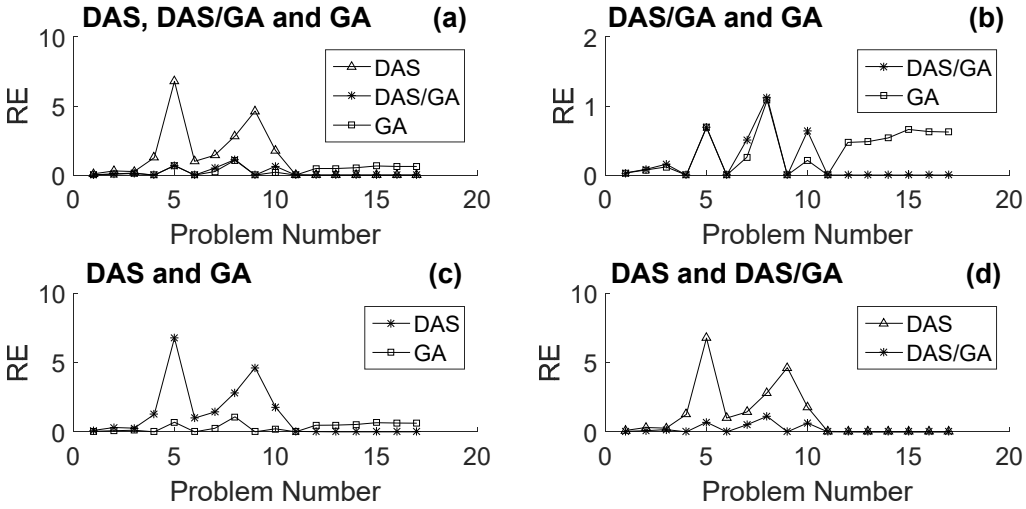

Figure 3. Comparison between the performances of the different methods using *RE* measure: (**a**) DAS/GA vs. DAS and GA, (**b**) DAS/GA vs. GA, (**c**) DAS vs. GA, and (**d**) DAS/GA vs. DAS.

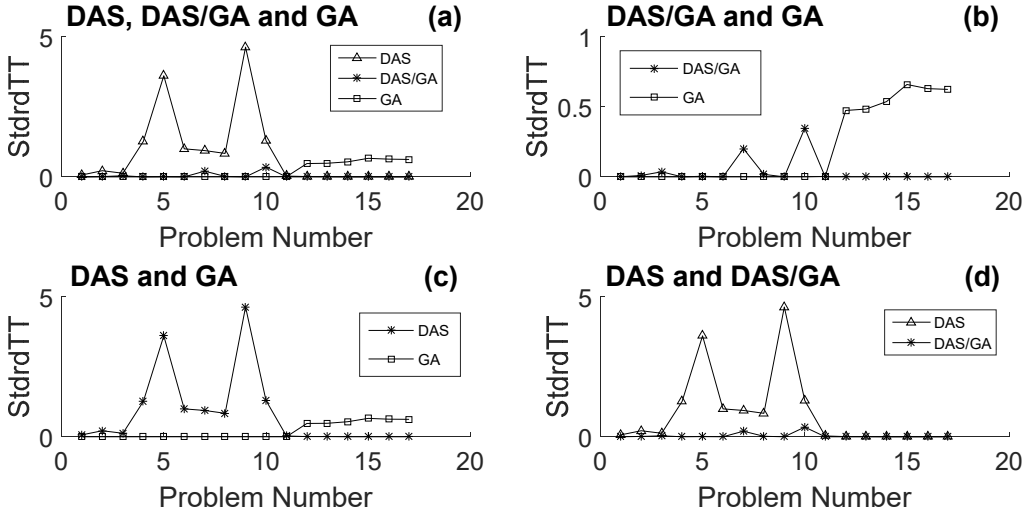

Figure 4. Comparison between the performances of the different methods using *StdrdTT* measure: (**a**) DAS/GA vs. DAS and GA, (**b**) DAS/GA vs. GA, (**c**) DAS vs. GA, and (**d**) DAS/GA vs. DAS.

The negligible difference in performance between DAS-*h* and DAS/GA shown in Figure 3d for large problems suggests that when combining DAS-*h* with GA to form the DAS/GA meta-heuristic, DAS/GA will be trapped in a region of good local optima created by DAS-*h;* consequently, the effect of GA will be negligible as it is trapped in this region.

In Figure 4, the standardized total tardiness values $StdrdTT_H$ for the different methods were compared. Figure 4a shows the exact same trends as found in Figure 3a between the performances of the different methods but using the $StdrdTT$ performance measure. Figure 4b shows that GA slightly outperformed DAS/GA for small problems and significantly for medium problems. For large problems, DAS/GA significantly outperformed GA, which is the same as the trend in Figure 3b. Figure 4c shows the same trend found in Figure 3c between the performances of GA and DAS-*h*.

Figure 4d shows that DAS/GA outperformed DAS-*h* for small problems and significantly for medium problems; however, the methods had a negligible difference in their performance for large problems, which is the same trend as found in Figure 4d.

Figure 5 shows a comparison between DAS/GA and GA using RAI measure. The figure reveals the same trend found earlier using RE and $StdrdTT$ measures in Figures 3b and 4b, respectively: GA slightly outperformed DAS/GA for small problems and especially for medium problems. For large problems, DAS/GA significantly outperformed GA. DAS-*h* helped GA to improve its performance significantly in large problems but deteriorated its performance for small and especially for medium problems, as the ARI values for DAS/GA for large problems are almost constant and approximately 1, but for small problems and especially for medium problems, the values are less than 1.

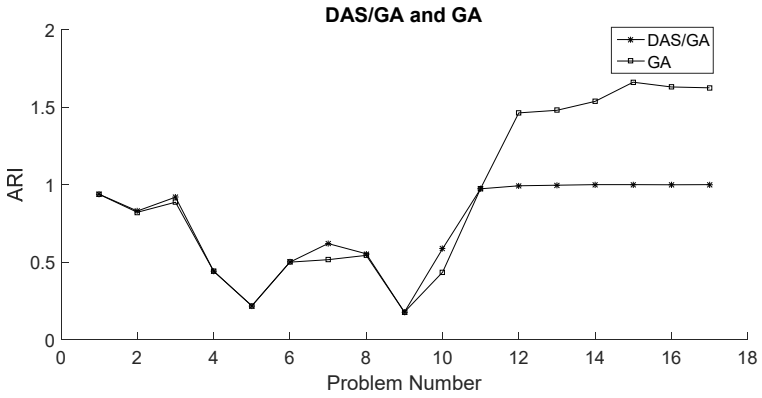

Figure 5. Comparison between DAS/GA and GA using ARI measure.

Moreover, looking at the ARI values for DAS/GA for large problems supports the argument made earlier: DAS-*h* creates a region of good local optima for large size problems that trap GA in it and renders the effect of GA negligible, and hence in this region DAS-*h* dictates the behavior of DAS/GA. The values of ARI support this argument, as the ARI values for large problems are almost 1, which indicates that the behavior of DAS/GA is very close to the behavior of DAS-*h*, and hence the effect of GA is almost negligible in DAS/GA compared to the effect of DAS-*h*.

Figure 6 compares the evolution line of DAS/GA and GA for problem 500_10_05_05. The Figure shows the same basic behavior for their evolution lines, except that DAS/GA's evolution line, shown in Figure 6b, started from a better point and reached a better fitness value than GA. Moreover, the figure shows that GA had a higher number of enhancement points in its evolution line than DAS/GA. This observation agrees with what was noted earlier about the effect of the region of local optima that DAS-*h* introduces in DAS/GA.

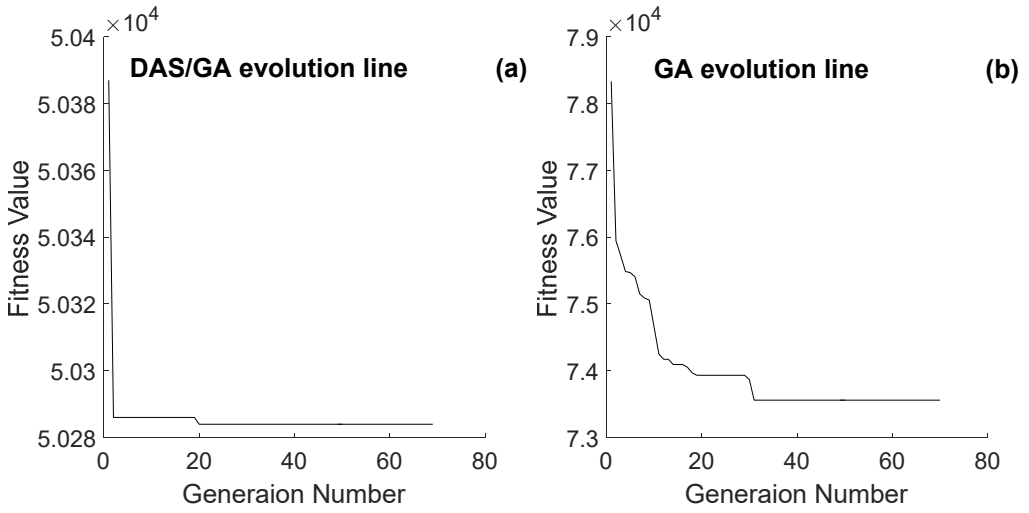

Figure 6. Evolution lines for (**a**) DAS/GA and (**b**) GA for problem 500_10_05_05.

To explore this observation more, Figure 7 shows a comparison between Average Number of Enhancements Per Replicate (ANEPR) for both meta-heuristics DAS/GA and GA. It is noted that as the problem size increased, the ANEPR for the GA method increased significantly, while for the DAS/GA method, the ANEPR was independent of the problem size, as the DAS/GA method showed a random relation between ANEPR and the problem size.

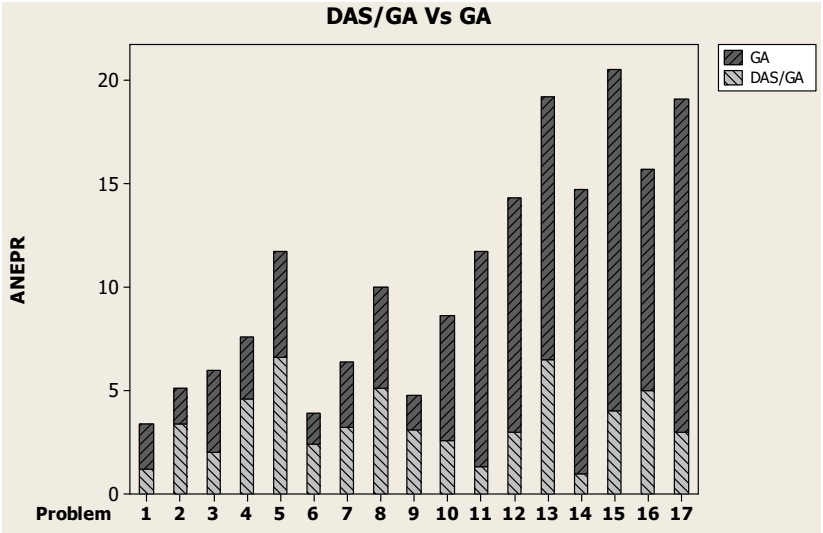

Figure 7. Bar graph for the number of enhancements for DAS/GA and GA.

This shows that generating a strong initial solution for DAS/GA for large-size problems using DAS-*h* will cause DAS/GA to be trapped in a region of local optima and hence limit the number of enhancements made to the strong initial solution. This matter makes the difference between the performances of DAS-*h* and DAS/GA negligible, as revealed in Figures 3d and 4d and the DAS/GA curve in Figure 5, and hence supports the argument made earlier that DAS-*h* dictates the behavior of DAS/GA in this region of large-size problems.

Figure 8 shows a comparison between the performance of ATC-*h* and DAS-*h*. The Figure shows that DAS-*h* outperformed ATC-*h* for most of the problems, whether they are small, medium, or large. Moreover, the Figure shows that the largest difference between the two heuristics is for the medium-size problems, where DAS-*h* significantly outperformed ATC-*h*.

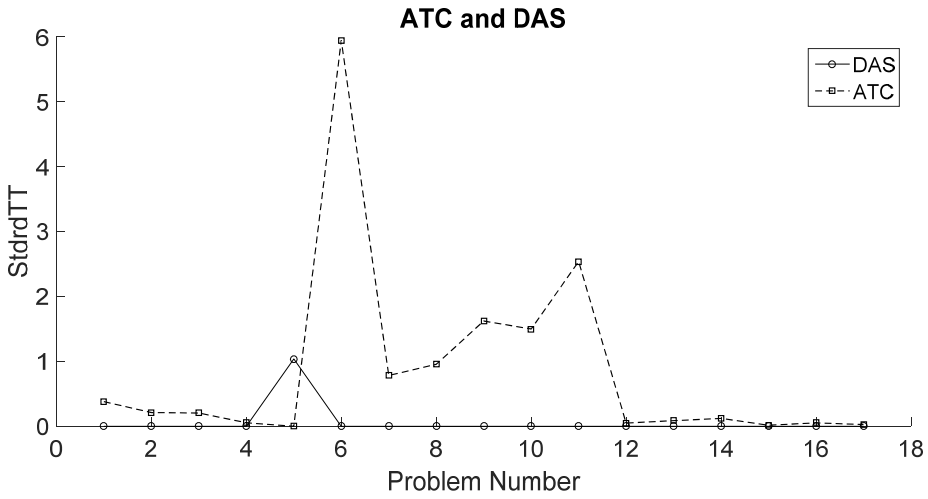

Figure 8. Comparison between ATC-*h* and DAS-*h* performance.

This superior performance of DAS-*h* can be linked to the structure of this heuristic, where DDT-*h* first constructs a good initial feasible schedule; then, this schedule is coarse-tuned using ATC-*h*, after which the S-*h* fine-tunes the schedule.

This superior performance of DAS-*h* in medium-range problems explains the strong deterioration in the performance of DAS/GA relative to GA in medium-size problems. DAS-*h* generates a region of local optima that traps DAS/GA, and consequently the performance of DAS/GA is limited in this region to the performance of DAS-*h*. Unlike the case with large-size problems, in medium-size problems GA alone can reach better regions than the region provided by DAS-*h* in DAS/GA; therefore, GA alone outperforms the performance of DAS/GA in medium-size problems, as the region provided by DAS-*h* traps DAS/GA in it and limits the capabilities of GA in DAS/GA to reach better regions.

6. Conclusions

This article proposed a Heuristic-Based Genetic Algorithm (DAS/GA) to minimize the overall total tardiness in scheduling large-size identical parallel machines with a single server. The results showed that DAS-*h* significantly enhanced the performance of GA for large-size problems where MIP failed due to high execution time and/or memory limitations, and GA performed badly due to the large search space. The results also showed that the usage of the proposed meta-heuristic DAS/GA algorithm should be limited only to large-size problems because DAS-*h* limited the capabilities of GA in small-size problems and especially in medium-size problems and because MIP performed well in small-size problems, which rendered the usage of DAS/GA unnecessary in this range of problems.

Future works will investigate existing meta-heuristic algorithms for scheduling large identical parallel machines with a single server.

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

Funding: This research received no external funding.

Data Availability Statement: The procedure for generating the data presented in this work was stated in the manuscript. However, it can be requested from the corresponding author.

Acknowledgments: The authors are grateful for the valuable comments raised by the reviewers aiming to enhance the quality of the manuscript.

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

References

1. 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]
2. Heydar, M.; Mardaneh, E.; Loxton, R. Approximate dynamic programming for an energy-efficient parallel machine scheduling problem. *Eur. J. Oper. Res.* **2022**, *302*, 363–380. [CrossRef]
3. Asadpour, M.; Hodaei, Z.; Azami, M.; Kehtari, E.; Vesal, N. A green model for identical parallel machines scheduling problem considering tardy jobs and job splitting property. *Sustain. Oper. Comput.* **2022**, *3*, 149–155. [CrossRef]
4. Vincent, B.; Duhamel, C.; Ren, L.; Tchernev, N. An efficient heuristic for scheduling on identical parallel machines to minimize total tardiness. *IFAC-PapersOnLine* **2016**, *49*, 1737–1742. [CrossRef]
5. Safarzadeh, H.; Niaki, S.T.A. Bi-objective green scheduling in uniform parallel machine environments. *J. Clean. Prod.* **2019**, *217*, 559–572. [CrossRef]
6. Bazargan-Lari, M.R.; Taghipour, S.; Zaretalab, A.; Sharifi, M. Correction to: Production scheduling optimization for parallel machines subject to physical distancing due to COVID-19 pandemic. *Oper. Manag. Res.* **2022**. [CrossRef]
7. Salimifard, K.; Li, J.; Mohammadi, D.; Moghdani, R. A multi objective volleyball premier league algorithm for green scheduling identical parallel machines with splitting jobs. *Appl. Intell.* **2021**, *51*, 4143–4161. [CrossRef]
8. Farmand, N.; Zarei, H.; Rasti-Barzoki, M. Two meta-heuristic algorithms for optimizing a multi-objective supply chain scheduling problem in an identical parallel machines environment. *Int. J. Ind. Eng. Comput.* **2021**, *12*, 249–272. [CrossRef]
9. Mahmud, S.; Abbasi, A.; Chakrabortty, R.K.; Ryan, M.J. A self-adaptive hyper-heuristic based multi-objective optimisation approach for integrated supply chain scheduling problems. *Knowl.-Based Syst.* **2022**, *251*, 109190. [CrossRef]
10. Wu, X.; Peng, J.; Xiao, X.; Wu, S. An effective approach for the dual-resource flexible job shop scheduling problem considering loading and unloading. *J. Intell. Manuf.* **2021**, *32*, 707–728. [CrossRef]
11. Allahverdi, A.; Mittenthal, J. Scheduling on M parallel machines subject to random breakdowns to minimize expected mean flow time. *Nav. Res. Logist.* **1994**, *41*, 677–682. [CrossRef]
12. Liao, C.-J.; Tsou, H.-H.; Huang, K.-L. Neighborhood search procedures for single machine tardiness scheduling with sequence-dependent setups. *Theor. Comput. Sci.* **2012**, *434*, 45–52. [CrossRef]
13. Wodecki, M. A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem. *Int. J. Adv. Manuf. Technol.* **2008**, *37*, 996–1004. [CrossRef]
14. Allali, K.; Aqil, S.; Belabid, J. Distributed no-wait flow shop problem with sequence dependent setup time: Optimization of makespan and maximum tardiness. *Simul. Model. Pract. Theory* **2022**, *116*, 102455. [CrossRef]
15. Yanıkoğlu, İ.; Yavuz, T. Branch-and-price approach for robust parallel machine scheduling with sequence-dependent setup times. *Eur. J. Oper. Res.* **2022**, *301*, 875–895. [CrossRef]
16. Vepsalainen, A.P.J.; Morton, T.E. Priority Rules for Job Shops with Weighted Tardiness Costs. *Manag. Sci.* **1987**, *33*, 947–1068. [CrossRef]
17. Lee, Y.H.; Pinedo, M. Scheduling jobs on parallel machines with sequence-dependent setup times. *Eur. J. Oper. Res.* **1997**, *100*, 464–474. [CrossRef]
18. Lee, Y.H.; Bhaskaran, K.; Pinedo, M. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Trans.* **1997**, *29*, 45–52. [CrossRef]
19. Morton, T.E.; Rachamadugu, R.M.V. *Myopic Heuristics for the Single Machine Weighted Tardiness Problem*; Carnegic-Mellon University: Pittsburgh, PA, USA, 1982.
20. Pfund, M.; Fowler, J.W.; Gadkari, A.; Chen, Y. Scheduling jobs on parallel machines with setup times and ready times. *Comput. Ind. Eng.* **2008**, *54*, 764–782. [CrossRef]
21. Biele, A.; Mönch, L. Decomposition methods for cost and tardiness reduction in aircraft manufacturing flow lines. *Comput. Oper. Res.* **2019**, *103*, 134–147. [CrossRef]
22. Schaller, J.; Valente, J.M.S. Heuristics for scheduling jobs in a permutation flow shop to minimize total earliness and tardiness with unforced idle time allowed. *Expert Syst. Appl.* **2019**, *119*, 376–386. [CrossRef]
23. Alves, F.F.; Nogueira, T.H.; Ravetti, M.G. Learning algorithms to deal with failures in production planning. *Comput. Ind. Eng.* **2022**, *169*, 108231. [CrossRef]
24. Balasubramanian, H.; Mönch, L.; Fowler, J.; Pfund, M. Genetic algorithm based scheduling of parallel batch machines with incompatible job families to minimize total weighted tardiness. *Int. J. Prod. Res.* **2004**, *42*, 1621–1638. [CrossRef]
25. Chaudhry, I.A.; Drake, P.R. Minimizing total tardiness for the machine scheduling and worker assignment problems in identical parallel machines using genetic algorithms. *Int. J. Adv. Manuf. Technol.* **2008**, *42*, 581. [CrossRef]
26. Xhafa, F.; Sun, J.; Barolli, A.; Biberaj, A.; Barolli, L. Genetic Algorithms for Satellite Scheduling Problems. *Mob. Inf. Syst.* **2012**, *8*, 717658. [CrossRef]
27. Belkadi, K.; Gourgand, M.; Benyettou, M. Parallel genetic algorithms with migration for the hybrid flow shop scheduling problem. *J. Appl. Math. Decis. Sci.* **2006**, *2006*, 65746. [CrossRef]

28. Costantino, F.; De Toni, A.F.; Di Gravio, G.; Nonino, F. Scheduling Mixed-Model Production on Multiple Assembly Lines with Shared Resources Using Genetic Algorithms: The Case Study of a Motorbike Company. *Adv. Decis. Sci.* **2014**, *2014*, 874031. [CrossRef]
29. Chen, C.; Yang, Z.; Tan, Y.; He, R. Diversity Controlling Genetic Algorithm for Order Acceptance and Scheduling Problem. *Math. Probl. Eng.* **2014**, *2014*, 367152. [CrossRef]
30. Karimi-Mamaghan, M.; Mohammadi, M.; Meyer, P.; Karimi-Mamaghan, A.M.; Talbi, E.-G. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *Eur. J. Oper. Res.* **2022**, *296*, 393–422. [CrossRef]
31. Wang, X.; Li, Z.; Chen, Q.; Mao, N. Meta-heuristics for unrelated parallel machines scheduling with random rework to minimize expected total weighted tardiness. *Comput. Ind. Eng.* **2020**, *145*, 106505. [CrossRef]
32. Khalid, O.W.; Isa, N.A.M.; Mat Sakim, H.A. Emperor penguin optimizer: A comprehensive review based on state-of-the-art meta-heuristic algorithms. *Alex. Eng. J.* **2023**, *63*, 487–526. [CrossRef]
33. Mensendiek, A.; Gupta, J.N.D.; Herrmann, J. Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. *Eur. J. Oper. Res.* **2015**, *243*, 514–522. [CrossRef]
34. Schaller, J.E. Minimizing total tardiness for scheduling identical parallel machines with family setups. *Comput. Ind. Eng.* **2014**, *72*, 274–281. [CrossRef]
35. Shim, S.-O.; Kim, Y.-D. Scheduling on parallel identical machines to minimize total tardiness. *Eur. J. Oper. Res.* **2007**, *177*, 135–146. [CrossRef]
36. Min, L.; Cheng, W. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. *Artif. Intell. Eng.* **1999**, *13*, 399–403. [CrossRef]
37. Kashan, A.H.; Karimi, B.; Jenabi, M. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Comput. Oper. Res.* **2008**, *35*, 1084–1098. [CrossRef]
38. Cochran, J.K.; Horng, S.-M.; Fowler, J.W. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Comput. Oper. Res.* **2003**, *30*, 1087–1102. [CrossRef]
39. Chang, P.-C.; Chen, S.-H.; Lin, K.-L. Two-phase sub population genetic algorithm for parallel machine-scheduling problem. *Expert Syst. Appl.* **2005**, *29*, 705–712. [CrossRef]
40. Leksakul, K.; Phetsawat, S. Nurse Scheduling Using Genetic Algorithm. *Math. Probl. Eng.* **2014**, *2014*, 246543. [CrossRef]
41. Lin, S.-W.; Ying, K.-C. A hybrid approach for single-machine tardiness problems with sequence-dependent setup times. *J. Oper. Res. Soc.* **2008**, *59*, 1109–1119. [CrossRef]
42. Laha, D. A Simulated Annealing Heuristic for Minimizing Makespan in Parallel Machine Scheduling. In Proceedings of the SEMCCO 2012: Swarm, Evolutionary, and Memetic Computing, Bhubaneswar, India, 20–22 December 2012; Panigrahi, B.K., Das, S., Suganthan, P.N., Nanda, P.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 198–205.
43. Liao, C.-J.; Juan, H.-C. An ant colony optimization for single-machine tardiness scheduling with sequence-dependent setups. *Comput. Oper. Res.* **2007**, *34*, 1899–1909. [CrossRef]
44. Anghinolfi, D.; Paolucci, M. A new ant colony optimization approach for the single machine total weighted tardiness scheduling prob-lem. *Int. J. Oper. Res.* **2008**, *5*, 44–60.
45. Jia, Z.; Yan, J.; Leung, J.Y.T.; Li, K.; Chen, H. Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities. *Appl. Soft Comput.* **2019**, *75*, 548–561. [CrossRef]
46. Tirkolaee, E.B.; Goli, A.; Hematian, M.; Sangaiah, A.K.; Han, T. Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms. *Computing* **2019**, *101*, 547–570. [CrossRef]
47. Mohamad Shirajuddin, T.; Muhammad, N.S.; Abdullah, J. Optimization problems in water distribution systems using Non-dominated Sorting Genetic Algorithm II: An overview. *Ain Shams Eng. J.* **2022**, 101932. [CrossRef]
48. Yusuf, A.; Bayhan, N.; Tiryaki, H.; Hamawandi, B.; Toprak, M.S.; Ballikaya, S. Multi-objective optimization of concentrated Photovoltaic-Thermoelectric hybrid system via non-dominated sorting genetic algorithm (NSGA II). *Energy Convers. Manag.* **2021**, *236*, 114065. [CrossRef]
49. Liu, Y.; You, K.; Jiang, Y.; Wu, Z.; Liu, Z.; Peng, G.; Zhou, C. Multi-objective optimal scheduling of automated construction equipment using non-dominated sorting genetic algorithm (NSGA-III). *Autom. Constr.* **2022**, *143*, 104587. [CrossRef]
50. Zhou, Y.; Zhang, W.; Kang, J.; Zhang, X.; Wang, X. A problem-specific non-dominated sorting genetic algorithm for supervised feature selection. *Inf. Sci.* **2021**, *547*, 841–859. [CrossRef]
51. Yazdani, M.; Zandieh, M.; Tavakkoli-Moghaddam, R. Evolutionary algorithms for multi-objective dual-resource constrained flexible job-shop scheduling problem. *OPSEARCH* **2019**, *56*, 983–1006. [CrossRef]
52. Sharma, S.; Chadha, M.; Kaur, H. Multi-step crossover genetic algorithm for bi-criteria parallel machine scheduling problems. *Int. J. Math. Oper. Res.* **2020**, *18*, 71–84. [CrossRef]
53. Li, K.; Yang, S. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Appl. Math. Model.* **2009**, *33*, 2145–2158. [CrossRef]
54. Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Recent Developments in Deterministic Sequencing and Scheduling: A Survey BT—Deterministic and Stochastic Scheduling. In *Proceedings of the Part of the NATO Advanced Study Institutes Series Book Series (ASIC, Volume 84)*; Dempster, M.A.H., Lenstra, J.K., Rinnooy Kan, A.H.G., Eds.; Springer: Dordrecht, The Netherlands, 1982; pp. 35–73.

55. Grefenstette, J.J.; Baker, J.E. How genetic algorithms work: A critical look at implicit parallelism. In Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, Fairfax, VA, USA, 4–7 June 1989; Schaffer, D., Ed.; Morgan Kaufmann Publishers Inc.: Fairfax, VA, USA, 1989; pp. 20–27.
56. Moscato, P.; Norman, M.G. A "Memetic" Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In Proceedings of the International Conference on Parallel Computing and Transputer Applications, Barcelona, Spain, 21–25 September 1992; Valero, M., Ed.; IOS Press: Barcelona, Spain, 1992; pp. 177–186.
57. Tanaka, S.; Araki, M. A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines. *Int. J. Prod. Econ.* **2008**, *113*, 446–458. [CrossRef]
58. Belouadah, H.; Potts, C.N. Scheduling identical parallel machines to minimize total weighted completion time. *Discret. Appl. Math.* **1994**, *48*, 201–218. [CrossRef]
59. Fisher, M.L. A dual algorithm for the one-machine scheduling problem. *Math. Program.* **1976**, *11*, 229–251. [CrossRef]