Mathematical and Computational Applications

MDPI

Article

Chaotic Multi-Objective Simulated Annealing and Threshold Accepting for Job Shop Scheduling Problem

Juan Frausto-Solis [1,*], Leonor Hernández-Ramírez [1], Guadalupe Castilla-Valdez [1], Juan J. González-Barbosa [1] and Juan P. Sánchez-Hernández [2]

[1] Graduate Program Division, Tecnológico Nacional de México/Instituto Tecnológico de Ciudad Madero, Cd. Madero 89440, Mexico; iscleo1@gmail.com (L.H.-R.); gpe_cas@yahoo.com.mx (G.C.-V.); jjgonzalezbarbosa@hotmail.com (J.J.G.-B.)

[2] Dirección de Informática, Electrónica y Telecomunicaciones, Universidad Politécnica del Estado de Morelos, Boulevard Cuauhnáhuac 566, Jiutepec 62574, Mexico; juan.paulosh@upemor.edu.mx

* Correspondence: juan.frausto@gmail.com

Abstract: The Job Shop Scheduling Problem (JSSP) has enormous industrial applicability. This problem refers to a set of jobs that should be processed in a specific order using a set of machines. For the single-objective optimization JSSP problem, Simulated Annealing is among the best algorithms. However, in Multi-Objective JSSP (MOJSSP), these algorithms have barely been analyzed, and the Threshold Accepting Algorithm has not been published for this problem. It is worth mentioning that the researchers in this area have not reported studies with more than three objectives, and the number of metrics they used to measure their performance is less than two or three. In this paper, we present two MOJSSP metaheuristics based on Simulated Annealing: Chaotic Multi-Objective Simulated Annealing (CMOSA) and Chaotic Multi-Objective Threshold Accepting (CMOTA). We developed these algorithms to minimize three objective functions and compared them using the HV metric with the recently published algorithms, MOMARLA, MOPSO, CMOEA, and SPEA. The best algorithm is CMOSA (HV of 0.76), followed by MOMARLA and CMOTA (with HV of 0.68), and MOPSO (with HV of 0.54). In addition, we show a complexity comparison of these algorithms, showing that CMOSA, CMOTA, and MOMARLA have a similar complexity class, followed by MOPSO.

Keywords: JSSP; CMOSA; CMOTA; chaotic perturbation

Citation: Frausto-Solis, J.; Hernández-Ramírez, L.; Castilla-Valdez, G.; González-Barbosa, J.J.; Sánchez-Hernández, J.P. Chaotic Multi-Objective Simulated Annealing and Threshold Accepting for Job Shop Scheduling Problem. *Math. Comput. Appl.* **2021**, *26*, 8. https://doi.org/10.3390/mca26010008

Received: 26 September 2020
Accepted: 8 January 2021
Published: 12 January 2021

1. Introduction

The Job Shop Scheduling Problem (JSSP) has enormous industrial applicability. This problem consists of a set of jobs, formed by operations, which must be processed in a set of machines subject to constraints of precedence and resource capacity. Finding the optimal solution for this problem is too complex, and so it is classified in the NP-hard class [1,2]. On the other hand, the JSSP foundations provide a theoretical background for developing efficient algorithms for other significant sequencing problems, which have many production systems applications [3]. Furthermore, designing and evaluating new algorithms for JSSP is relevant not only because it represents a big challenge but also for its high industrial applicability [4].

There are several JSSP taxonomies; one of which is single-objective and multi-objective optimization. The single-objective optimization version has been widely studied for many years, and the Simulated Annealing (SA) [5] is among the best algorithms. The Threshold Accepting (TA) algorithm from the same family is also very efficient in this area [6]. In contrast, in the case of Multi-Objective Optimization Problems (MOOPs), both algorithms for JSSP and their comparison are scarce.

Published JSSP algorithms for MOOP include only a few objectives, and only a few performance metrics are reported. However, it is common for the industrial scheduling requirements to have several objectives, and then the Multi-Objective JSSP (MOJSSP) becomes an even more significant challenge. Thus, many industrial production areas require the multi-objective approach [7,8].

In single-objective optimization, the goal is to find the optimal feasible solution of an objective function. In other words, to find the best value of the variables which fulfill all the constraints of the problem. On the other hand, for MOJSSP, the problem is to find the optimum of a set of objective functions $f_1(x)$, $f_2(x) \ldots f_n(x)$ depending on a set of variables x and subject to a set of constraints defined by these variables. To find the optimal solution is usually impossible because fulfilling some objective functions may not optimize the other objectives of the problem. In MOOP, a preference relation or Pareto dominance relation produces a set of solutions commonly called the Pareto optimal set [9]. The Decision Makers (DMs) should select from the Pareto set the solution that satisfies their preferences, which can be subjective, based on experience, or will most likely be influenced by the industrial environment's needs [10]. Therefore, the DM needs to have a Pareto front that contains multiple representative compromise solutions, which exhibit both good convergence and diversity [11].

In the study of single-objective JSSP, many algorithms have been applied. Some of the most common are SA, Genetic Algorithms (GAs), Tabu Search (TS), and Ant Systems (ASs) [12]. In addition, as we mention below, few works in the literature solve JSSP instances with more than two objectives and applying more than two metrics to evaluate their performance. Nevertheless, for MOJSSP, the number of objectives and performance metrics remains too small [8,13–15]. The works of Zhao [14] and Mendez [8] are exceptions because the authors have presented implementations with two or three significant objective functions and two performance metrics. Moreover, SA and TA have shown to be very efficient for solving NP-hard problems. Thus, this paper's motivation is to develop new efficient SA algorithms for MOJSSP with two or more objective functions and a larger number of performance metrics.

The first adaptation of SA to MOOP was an algorithm proposed in 1992, also known as MOSA [16]. An essential part of this algorithm is that it applies the Boltzmann criterion for accepting bad solutions, commonly used in single-objective JSSP. MOSA combines several objective functions. The single-objective JSSP optimization with SA algorithm and MOSA algorithm for multi-objective optimization is different in several aspect related to determining the energy functions, using and generating new solutions, and measuring their quality as is well known, these energy functions are required in the acceptance criterion. Multiple versions of MOSA have been proposed in the last few years. One of them, published in 2008, is AMOSA, that surpassed other MOOP algorithms at this time [17]. In this work, we adapt this algorithm for MOJSSP. TA [6] is an algorithm for single-objective JSSP, which is very similar to Simulated Annealing. These two algorithms have the same structure, and both use a temperature parameter, and they accept some bad solutions for escaping from local optima. In addition, these algorithms are among the best JSSP algorithms, and their performance is very similar. Nevertheless, for MOJSSP, a TA algorithm has not been published, and so for obvious reason, it was not compared with the SA multi-objective version.

MOJSSP has been commonly solved using IMOEA/D [14], NSGA-II [18], SPEA [19], MOPSO [20], and CMOEA [21]; the latter was renamed CMEA in [8]. Nevertheless, the number of objectives and performance metrics of these algorithms remains too small. The Evolutionary Algorithm based on decomposition proposed in 2016 by Zhao in [14] was considered the best algorithm [22]. The Multi-Objective Q-Learning algorithm (MOQL) for JSSP was published in 2017 [23]; this approach uses several agents to solve JSSP. An extension of MOQL is MOMARLA, which was proposed in 2019 by Mendez [8]. This MOJSSP algorithm uses two objective functions: makespan and total tardiness. MOMARLA overcomes the classical multi-objective algorithms SPEA [19], CMOEA [21], and MOPSO [20].

The two new algorithms presented in this paper for JSSP are Chaotic Multi-Objective Simulated Annealing (CMOSA) and Chaotic Multi-Objective Threshold Accepting (CMOTA). The first algorithm is inspired by the classic MOSA algorithm [17]. However, CMOSA is different in three aspects: (1) for the first time it is designed specifically for MOJSSP, (2) it uses an analytical tuning of the cooling scheme parameters, and (3) it uses chaotic perturbations for finding new solutions and for escaping from local optima. This process allows the search to continue from a different point in the solution space and it contributes to a better diversity of the generated solutions. Furthermore, CMOTA is based on CMOSA and Threshold Accepting, and it does not require the Boltzmann distribution. Instead, it uses a threshold strategy for accepting bad solutions to escape from local optima. In addition, a chaotic perturbation function is applied.

In this paper, we present two new alternatives for MOJSSP, and we consider three objective functions: makespan, total tardiness, and total flow time. The first objective is very relevant for production management applications [7], while the other two are critical for enhancing client attention service [23]. In addition, we use six metrics for the evaluation of these algorithms, and they are Mean Ideal Distance (MID), Spacing (S), Hypervolume (HV), Spread (Δ), Inverted Generational Distance (IGD), and Coverage (C). We also apply an analytical tuning parameter method to these algorithms. Finally, we compare the achieved results with those obtained with the JSSP algorithm cited below in [8,14].

The rest of the paper is organized as follows. In Section 2, we make a qualitative comparison of related MOJSSP works. In Section 3, we present MOJSSP concepts and the performance metrics that were applied. Section 4 presents the formulation of MOJSSP with three objectives. The proposed algorithms, their tuning method, and the chaotic perturbation are also shown in Section 5. Section 6 shows the application of the proposed algorithms to a set of 70, 58, and 15 instances. Finally, the results are shown and compared with previous works. In Section 7, we present our conclusions.

2. Related Works

As mentioned above, in single-objective optimization, the JSSP community has broadly investigated the performance of the different solution methods. However, the situation is entirely different for MOJSSP, and there is a small number of published works. In 1994, an analysis of SA family algorithms for JSSP was presented [24]; two of them were SA and TA, which we briefly explain in the next paragraph. These algorithms suppose that the solutions define a set of macrostates of a set of particles, while the objective functions' values represent their energy, and both algorithms have a Metropolis cycle where the neighborhood of solutions is explored. In single-objective optimization, for the set of instances used to evaluate JSSP algorithms, SA obtained better results than TA. Furthermore, a better solution than the previous one is always accepted, while a worse solution may be accepted depending on the Boltzmann distribution criterion. This distribution is related to the current temperature value and the increment or decrement of energy (associated with the objective functions) in the current temperature value. In the TA case, a worse solution than the previous one may be accepted using a criterion that tries to emulate the Boltzmann distribution. This criterion establishes a possible acceptance of a worse solution when the decrement of energy is smaller than a threshold value depending on the temperature and a parameter γ that is very close to one. Then at the beginning of the process, the threshold values are enormous because they depend on the temperatures. Subsequently, the temperature parameter is gradually decreased until a value close to zero is achieved, and then this threshold is very small.

In 2001, a Multi-Objective Genetic Algorithm was proposed to minimize the makespan, total tardiness, and the total idle time [25]. The proposed methodology for JSSP was assessed with 28 benchmark problems. In this publication, the authors randomly weighted the different fitness functions to determine their results.

In 2006, SA was used for two objectives: the makespan and the mean flow time [26]. This algorithm was called Pareto Archived Simulated Annealing (PASA), which used the Simulated Annealing algorithm with an overheating strategy to escape from local optima and to improve the quality of the results. The performance of this algorithm was evaluated with 82 instances taken from the literature. Unfortunately, this method has not been updated for three or more objective functions.

In 2011, a two-stage genetic algorithm (2S-GA) was proposed for JSSP with three objectives to minimize the makespan, total weighted earliness, and total weighted tardiness [13]. In the first stage, a parallel GA found the best solution for each objective function. Then, in the second stage, the GA combined the populations, which evolved using the weighted aggregating objective function.

Researchers from the Contemporary Design and Integrated Manufacturing Technology (CDIMT) laboratory proposed an algorithm named Improved Multi-Objective Evolutionary Algorithm based on Decomposition (IMOEA/D) to minimize the makespan, tardiness, and total flow time [14]. The authors experiment with 58 benchmark instances, and they use the performance metrics Coverage [27] and Mean Ideal Distance (MID) [28] to evaluate their algorithm. We notice in Table 1, studies with two or three objectives, but they do not report any metric. On the other hand, IMOEA/D stands out from the rest of the literature, not only because the authors reported good results but also because they considered a more significant number of objectives, and they applied two metrics.

In 2008, the AMOSA algorithm based on SA for several objectives was proposed [17]. In this paper, the authors reported that the AMOSA algorithm performed better than some MOEA algorithms, one of them NSGA-II [29]. They presented the main Boltzmann rules for accepting bad solutions. Unfortunately, a MOJSSP with AMOSA and with more than two objectives has not been published.

In 2017, a hybrid algorithm between an NSGA-II and a linear programming approach was proposed [15]; it was used to solve the FT10 instance of Taillard [30]. This algorithm minimized the weighted tardiness and energy costs. To evaluate the performance, the authors only used the HV metric.

In 2019, MOMARLA was proposed, a new algorithm based on Q-Learning to solve MOJSSP [8]. This work provided flexibility to use decision-maker preferences; each agent represented a specific objective and used two action selection strategies to find a diverse and accurate Pareto front. In Table 1, we present the last related studies for MOJSSP and the proposed algorithms.

This paper analyzes our algorithms CMOSA and CMOTA, as follows: (a) comparing CMOSA and CMOTA versus IMOEA/D [14], (b) comparing our algorithms with the results published for MOMARLA, MOPSO, CMOEA, and SPEA, and (c) comparing CMOSA versus CMOTA.

Table 1. Related Works.

Algorithm	Objectives	Metrics
SA [16]	Makespan	*
SA and TA [24]	Makespan	*
Hybrid GA [25]	Makespan, total tardiness, and total idle time	*
PASA [26]	Makespan, mean flow time	*
2S-GA [13]	Makespan, total weighted earliness, and total weighted tardiness	*
IMOEA/D [14]	Makespan, total flow time, and tardiness time	C, MID
Hybrid GA/LS/LP [15]	Weighted tardiness, and energy costs	HV
MOMARLA [8]	Makespan, total tardiness	HV
CMOSA and CMOTA (This paper)	Makespan, total tardiness, and total flow time	MID, S, HV, Δ, IGD and C

* Not reported.

3. Multi-Objective Optimization

In a single-objective problem, the algorithm finishes its execution when it finds the solution that optimizes the objective function or a very close optimal solution. However, for Multi-Objective Optimization, the situation is more complicated since several objectives must be optimized simultaneously. Then, it is necessary to find a set of solutions optimizing each of the objectives individually. These solutions can be contrasting because we can obtain the best solution for an objective function that is not the best for other objective functions.

3.1. Concepts

Definitions of some concepts of Multi-Objective Optimization are shown below.

Pareto Dominance: In general, for any optimization problem, solution A dominates another solution B if the following conditions are met [31]: A is strictly better than B on at least one objective, and A is not worse than B for any objective function.

Non-dominated set: In a set of P solutions, the non-dominated solutions P1 is integrated by solutions that accomplish the following conditions [31]: any pair of P1 solutions must be non-dominated (one regarding the other), and any solution that does not belong to P1 is dominated by at least one member of P1.

Pareto optimal set: The set of non-dominated solutions of the total search space.

Pareto front: The graphic representation of the non-dominated solutions of the multi-objective optimization problem.

3.2. Performance Metrics

In an experimental comparison of different optimization techniques or algorithms, it is always necessary to have the notion of performance. In the case of Multi-Objective Optimization, the definition of quality is much more complicated than for single-objective optimization problems because the multi-objective optimization criteria itself consists of multiple objectives, of which, the most important are:

1. To minimize the distance of the resulting non-dominated set to the true Pareto front.
2. To achieve an adequate distribution (for instance, uniform) of the solutions is desirable.
3. To maximize the extension of the non-dominated front for each of the objectives. In other words, a wide range of values must be covered by non-dominated solutions.

In general, it is difficult to find a single performance metric that encompasses all of the above criteria. In the literature, a large number of performance metrics can be found. The most popular performance metrics were used in this research and are described below:

Mean Ideal Distance: Evaluates the closeness of the calculated Pareto front (PF_{calc}) solutions with an ideal point, which is usually (0, 0) [28].

$$MID = \frac{\sum_{i=1}^{Q} c_i}{Q} \tag{1}$$

where $c_i = \sqrt{f_{1,i}^2 + f_{2,i}^2 + f_{3,i}^2}$ and $f_{1,i}, f_{2,i}, f_{3,i}$ are the values of the i-th non-dominated solution for their first, second, and third objective function, and Q is the number of solutions in the PF_{calc}.

Spacing: Evaluates the distribution of non-dominated solutions in the PF_{calc}. When several algorithms are evaluated with this metric, the best is that with the smallest S value [32].

$$S = \sqrt{\frac{\sum_{i=1}^{Q} (d_i - \bar{d})^2}{Q}} \tag{2}$$

where d_i measures the distance in the space of the objective functions between the i-th solution and its nearest neighbor; that is the j-th solution in the PF_{calc} of the algorithm, Q is the number of the solutions in the PF_{calc}, $\bar{d}$ is the average of the d_i, that is $\bar{d} = \sum_{i=1}^{Q} \frac{d_i}{Q}$ and $d_i = min_j(|f_1^i(x) - f_1^j(x)| + |f_2^i(x) - f_2^j(x)| + \cdots + |f_M^i(x) - f_M^j(x)|)$, where f_1^i, f_2^i are the values of the i-th non-dominated solution for their first and second objective function, f_1^j, f_2^j are the values of the j-th non-dominated solution for their first and second objective function respectively, M is the number of objective functions and $i, j = 1, \ldots Q$.

Hypervolume: Calculates the volume in the objective space that is covered by all members of the non-dominated set [33]. The HV metric is measured based on a reference point (W), and this can be found simply by constructing a vector with the worst values of the objective function.

$$HV = volume\left(\cup_{i=1}^{|Q|} v_i\right) \tag{3}$$

where v_i is a hypercube and is constructed with a reference point W and the solution i as the diagonal corners of the hypercube [31]. An algorithm that obtains the largest HV value is better. The data should be normalized by transforming the value in the range [0, 1] for each objective separately to perform the calculation.

Spread: This metric was proposed to have a more precise coverage value and considers the distance to the (extreme points) of the true Pareto front (PF_{true}) [29].

$$\Delta = \frac{\sum_{k=1}^{M} d_k^e + \sum_{i=1}^{Q} |d_i - \bar{d}|}{\sum_{k=1}^{M} d_k^e + Q \times \bar{d}} \tag{4}$$

where d_k^e measures the distance between the "extreme" point of the PF_{true} for the k-th objective function, and the nearest point of PF_{calc}, d_i corresponds to the distance between the solution i-th of the PF_{calc}, while its nearest neighbor, $\bar{d}$ corresponds to the average of the d_i and M is the number of objectives.

Inverted Generational Distance: It is an inverted indicator version of the Generational Distance (GD) metric, where all the distances are measured from the PF_{true} to the PF_{calc} [1].

$$IGD(Q) = \frac{\left(\sum_{j=1}^{|T|} \hat{d}_j^p\right)^{1/p}}{|T|} \tag{5}$$

where $T = \{t_1, t_2, \ldots, t_{|T|}\}$ that is, the solutions in the PF_{true} and $|T|$ is the cardinality of T, p is an integer parameter, in this paper $p = 2$ and $\hat{d}_j$ is the Euclidean distance from t_j to its nearest objective vector q in Q, according to (6).

$$d_j = \min_{q=1}^{|Q|} \sqrt{\sum_{m=1}^{M} (fm(t_j) - fm(q))^2} \tag{6}$$

where $fm(t)$ is the m-th objective function value of the t-th member of T and M is the number of objectives.

Coverage: Represents the dominance between set A and set B [27]. It is the ratio of the number of solutions in set B that were dominated by solutions in set A and the total number of solutions in set B. The C metric is defined by (7).

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|} \tag{7}$$

When $C(A, B) = 1$, all B solution are dominated or equal to solutions in A. Otherwise, $C(A, B) = 0$, represents situations in which none of the solutions in B is dominated by any solution in A. The higher the value of $C(A, B)$, the more solutions in B are dominated by solutions in A. Both $C(A, B)$ and $C(B, A)$ should be considered, since $C(B, A)$ is not necessarily equal to $1 - C(A, B)$.

4. Multi-Objective Job Shop Scheduling Problem

In JSSP, there are a set of n different jobs consisting of operations that must be processed in m different machines. There are a set of precedence constraints for these operations, and there are also resource capacity constraints for ensuring that each machine should process only one operation at the same time. The processing time of each operation is known in advance. The objective of JSSP is to determine the sequence of the operations in each machine (the start and finish time of each operation) to minimize certain objective functions subject to the constraints mentioned above. The most common objective is the makespan, which is the total time in which all the problem operations are processed. Nevertheless, real scheduling problems are multi-objective, and several objectives should be considered simultaneously.

The three objectives that are addressed in the present paper are:

Makespan: the maximum time of completion of all jobs.

Total tardiness: it is calculated as the total positive differences between the makespan and the due date of each job.

Total flow time: it is the summation of the completion times of all jobs.

The formal MOJSSP model can be formulated as follows [34,35]:

$$Optimize\ F(x) = [f_1(x), f_2(x), \ldots, f_q(x)]\ Subject\ to : x \in S \tag{8}$$

where q is the number of objectives, x is the vector of decision variables, and S represents the feasible region. Defined by the next precedence and capacity constraints, respectively:

$$\begin{array}{ll} t_j \geq t_i + p_i & \text{For all } i, j \in O \text{ when } i \text{ precedes } j \\ t_j \geq t_i + p_i \text{ or } t_i \geq t_j + p_j & \text{For all } i, j \in O \text{ when } M_i = M_j \end{array}$$

where:

t_i, t_j are the starting times for the jobs $i, j \in J$.
p_i and p_j are the processing times for the jobs $i, j \in J$.
$J : \{J_1, J_2, J_3, \ldots, J_n\}$ it is the set of jobs.
$M : \{M_1, M_2, M_3, \ldots M_m\}$ it is the set of machines.
O is the set of operations $O_{j,i}$ (operation i of the job j).

The objective functions of makespan, total tardiness, and total flow time, are defined by Equations (9)–(11), respectively.

$$f_1 = min\left(\max_{j=1}^{n} C_j\right) \tag{9}$$

where C_j is the makespan of job j.

$$f_2 = min\left(\sum_{j=1}^{n} T_j\right) = min\left(\sum_{j=1}^{n} max(0, C_j - D_j)\right) \tag{10}$$

where $T_j = max(0, C_j - D_j)$ is the tardiness of job j, and D_j is the due date of job j and is calculated with $D_j = \tau \sum_{i=1}^{m} p_{j,i}$ [36], where $p_{j,i}$ is the time required to process the job j in the machine i. In this case, the due date of the j job is the sum of the processing time of all its operations on all machines, multiplied by a narrowing factor (τ), which is in the range $1.5 \leq \tau \leq 2.0$ [14,36].

$$f_3 = min\sum_{j=1}^{n} C_j \tag{11}$$

5. Multi-Objective Proposed Algorithms

The two multi-objective algorithms presented in this section for solving JSSP are Chaotic Multi-Objective Simulated Annealing and Chaotic Multi-Objective Threshold Accepting. We describe these algorithms in this section after analyzing the single-objective optimization algorithms for JSSP.

5.1. Simulated Annealing

The algorithm SA proposed by Kirkpatrick et al. comes from a close analogy with the metal annealing process [5]. This process consists of heating and progressively cooling metal. As the temperature decreases, the molecules' movement slows down and tends to adopt a lower energy configuration. Kirkpatrick et al. proposed this algorithm for combinatorial optimization problems and to escape from local minima. It starts with an initial solution and generates a new solution in its neighborhood. If the new solution is better than the old solution, then it is accepted. Otherwise, SA applies the Boltzmann distribution, which determines if a bad solution can be taken as a strategy for escaping from local optima. This process is repeated many times until an equilibrium condition is accomplished.

The SA algorithm is shown in Algorithm 1. Line 1 receives the parameters: the initial ($T_{initial}$) and final (T_{final}) temperatures, the alpha value (α) for decreasing the temperature, and beta (β) for increasing the length of the Metropolis cycle. The current temperature (T_k) is set in line 2. An initial solution ($s_{current}$) is generated randomly in line 3. The stop criterion is evaluated (line 4); this main cycle is repeated while the current temperature (T_k) is higher than the final temperature (T_{final}). The Metropolis cycle starts in line 5, where a neighboring solution (s_{new}) is generated (line 6). In line 7 the increment ΔE of the objective function is determined for the current solution ($s_{current}$) and the new one (s_{new}). When this increment is negative (line 8) the new solution is the best. In this case, the new solution replaces the current solution (line 9). Otherwise, the Boltzmann criterion is applied (lines 11 and 12). This criterion allows the algorithm to escape from local optima depending on the current temperature and delta values. Finally, line 16 increases the number of iterations of the Metropolis cycle, and in line 17, the cooling function is applied to reduce the current temperature.

Algorithm 1 Classic Simulated Annealing algorithm

1: **procedure** SA($T_{initial}, T_{final}, \alpha, \beta, L_k$)
2: $T_k \leftarrow T_{initial}$
3: $s_{current} \leftarrow RandomInitialSolution()$
4: **while** $T_k \geq T_{final}$ **do**
5: **for** 1 to L_k **do**
6: $s_{new} \leftarrow perturbation(s_{current})$
7: $\Delta E \leftarrow E(s_{new}) - E(s_{current})$
8: **if** $\Delta E < 0$ **then**
9: $s_{current} \leftarrow s_{new}$
10: **else**
11: **if** $(e^{-\Delta E/T_k} > random(0,1)$ **then**
12: $s_{current} \leftarrow s_{new}$
13: **end if**
14: **end if**
15: **end for**
16: $L_k \leftarrow \beta \times L_k$
17: $T_k \leftarrow \alpha \times T_k$
18: **end while**
19: **return** $s_{current}$
20: **end procedure**

5.2. Analytical Tuning for Simulated Annealing

The parameters tuning process for the SA algorithm used in this paper is based on a method proposed in [37]. This method establishes that both the initial and final temperatures are functions of the maximum and minimum energy values E_{max} and E_{min}, respectively. These energies appeared in the Boltzmann distribution criterion that states that a bad solution is accepted in a temperature T when $random(0,1) \leq e^{-\Delta E/T}$. For JSSP, ΔE is obtained with the makespan. For this tuning method, these two functions are obtained from the neighborhood of different solutions randomly generated. A set of previous SA executions must be carried out for obtaining ΔE_{max} and ΔE_{min}. These value are used in the Boltzmann distribution for determining the initial and final temperatures. Then, the other parameters of Metropolis cycle are determined. The process used is detailed in the next paragraph.

Initial temperature ($T_{initial}$): It is the temperature value from which the search process begins. The probability of accepting a new solution is almost 1 at high temperatures so, its cost of deterioration is maximum. The initial temperature is associated with the maximum allowed deterioration and its defined acceptance probability. Let us define s_i as the current solution, s_j a new proposed solution, $E_{(s_i)}$ and $E_{(s_j)}$ are its associated costs, the maximum and minimum deterioration are ΔE_{max} and ΔE_{min}. Then $P(\Delta E_{max})$, is the probability of accepting a solution with the maximum deterioration and it is calculated with (12). Thus the value of the initial temperature $(T_{initial})$ is calculated with (13).

$$P(\Delta E_{max}) = e^{(\Delta E_{max}/T_{initial})} \tag{12}$$

$$T_{initial} = \frac{-\Delta E_{max}}{\ln(P(\Delta E_{max}))} \tag{13}$$

Final temperature (T_{final}): It is the temperature value at which the search stops. In the same way, the final temperature is determined with (14) according to the probability $P(\Delta E_{min})$, which is the probability of accepting a solution with minimum deterioration.

$$T_{final} = \frac{-\Delta E_{min}}{\ln(P(\Delta E_{min}))} \tag{14}$$

Alpha value (α): It is the temperature decrease factor. This parameter determines the speed at which the decrease in temperature will occur, for fast decrements 0.7 it is usually used and for slow decrements 0.99.

Cooling scheme: This function specifies how the temperature is decreased. In this case, the value of the current temperature (T_k) follows the geometric scheme (15).

$$T_{k+1} = \alpha T_k \tag{15}$$

Length of the Markov chain or iterations in Metropolis cycle (L_k): This refers to the number of iterations of the Metropolis cycle that is performed at each temperature k, this number of iterations can be constant or variable. It is well known that at high temperatures, only a few iterations are required since the stochastic equilibrium is rapidly reached [37]. However, at low temperatures, a much more exhaustive level of exploration is required. Thus, a larger L_k value must be used. If L_{min} is the value of L_k at the initial temperature, and L_{max} is the L_k at the final temperature, then the Formula (16) is used.

$$L_{k+1} = \beta L_k \tag{16}$$

where β is the increment coefficient of L_k. Since the Functions (15) and (16) are applied successively in SA from the initial to the final temperature, T_{final} and L_{max} are calculated with (17) and (18).

$$T_{final} = \alpha^n T_{initial} \tag{17}$$

$$L_{max} = \beta^n L_{min} \tag{18}$$

In (17) and (18) n is the number of steps from $T_{initial}$ to T_{final}, then (19) and (20) are obtained.

$$n = \frac{\ln(T_{final}) - \ln(T_{initial})}{\ln(\alpha)} \tag{19}$$

$$\beta = e^{\left(\frac{\ln(L_{max}) - \ln(L_{min})}{n}\right)} \tag{20}$$

The probability of selecting the solution s_j from N random samples in the neighborhood V_{si} is given by (21); and from this equation, the N value is obtained in (22), where the exploration level C is defined in Equation (23).

$$P(S_j) = 1 - e^{\frac{-N}{|V_{si}|}} \tag{21}$$

$$N = - \mid V_{si} \mid \ln(1 - P(S_j)) = C \mid V_{si} \mid \tag{22}$$

$$C = \ln(P(S_j)) \tag{23}$$

The length of the Markov chain or iterations of the Metropolis cycle are defined by (24).

$$L_{max} = N = C \mid V_{si} \mid \tag{24}$$

To guarantee a good exploration level, the C value determined by (23) must be established between $1 \leq C \leq 4.6$ [38].

5.3. Chaotic Multi-Objective Simulated Annealing (CMOSA)

As we previously mentioned, the AMOSA algorithm was proposed in [17]. However, this algorithm is designed for general purposes. In this work, we adapt the AMOSA for JSSP to include the following features: (1) the mathematical constraints of MOJSSP, and (2) the objective functions makespan, total tardiness, and total flow time.

CMOSA has the same features previously described and has the next three elements: (1) a new structure, (2) chaotic perturbation, and (3) apply dominance to select solutions. These elements are described in the next subsections.

5.3.1. CMOSA Structure

The CMOSA algorithm uses a chaotic phase to improve the quality of the solutions considering the three objectives. Algorithm 2 receives its parameters in line 1: initial temperature ($T_{initial}$), final temperature (T_{final}), alpha (α), beta (β), Metropolis iterations in every cycle (L_k), and the initial solution ($s_{current}$) to be improved. In lines 2 and 3, the variables of the algorithm are initialized. In line 4, the $s_{current}$ is processed to obtain the values for each of the three objectives as output. In line 5, the initial temperature is established as the current temperature (T_k). Then the main cycle begins in line 6. This cycle is repeated as long as the current temperature is greater than, or equal to, the final temperature. In line 7, the Metropolis cycle begins. Subsequently, the algorithm verifies if it is stagnant in line 8. If that is the case, lines 9 to 20 are executed. The number of iterations to perform a local search is established in line 10; this value is based on the number of tasks of the instance multiplied by an experimentally tuned parameter (in this case, this parameter is $timesLS = 10$).

In line 11, a local search begins. In the first iteration of this search, a chaotic perturbation (explained in Algorithm 4) is applied to the $s_{current}$ (line 12) to restart the search process from another point in the solution space. In further iterations, a regular perturbation is applied (line 14) that consists only of exchanging the position of two operations in the solution, always verifying that the solution generated is feasible. In line 16, the s_{new} is processed to obtain the values for each of the three objectives. Subsequently, and only if the new solution dominates the current solution of the three objectives, the new solution is used to continue the search process (lines 17 and 18). When the algorithm is not stagnant, a regular perturbation is applied, and the flow continues (line 22). If the current and the new solution are different, we proceed with the dominance verification process to determine which solution is used to continue the search (line 26); this process is explained in Algorithm 5. Finally, from lines 29 to 36, a process is applied to set a limit to the number of times the algorithm is stagnant (See Algorithm 3). The algorithm is determined to be stagnant if, after some iterations, it fails to generate a new, non-dominated solution. In this algorithm, the stagnation is limited to 10 iterations. At the end of the algorithm, in line 37, the number of repetitions of the Metropolis cycle (L_k) is increased by multiplying its previous value by the β parameter value. Additionally, in line 38, the current temperature (T_k) is decreased by multiplying it by the α value. At the end of line 40, the stored solution ($s_{current}$) is generated as the output of the algorithm.

Algorithm 2 Chaotic Multi-Objective Simulated Annealing (CMOSA)

1: **procedure** CMOSA($T_{initial}, T_{final}, \alpha, \beta, L_k, s_{current}$)
2: $MAXSTAGNANT \leftarrow 10, counterTrapped \leftarrow 0, isCaught \leftarrow FALSE$
3: $iterationsLocalSearch \leftarrow tasks \times timesLS, verifyCaught \leftarrow TRUE, countCaught \leftarrow 0$
4: $mks_{current}, tds_{current}, flt_{current} \leftarrow calculateValues(s_{current})$ ▷ $mks : makespan, tds : tardiness, flt : flowtime$
5: $T_k \leftarrow T_{initial}$
6: **while** $T_k \geq T_{final}$ **do**
7: **for** $i \leftarrow 0$ to L_k **do**
8: **if** $isCaught = TRUE$ **then**
9: $isCaught \leftarrow FALSE$
10: **for** $k \leftarrow 0$ to $iterationsLocalSearch$ **do**
11: **if** $k = 0$ **then**
12: $s_{new} \leftarrow chaoticPerturbation(s_{current})$ ▷ See Algorithm 4
13: **else**
14: $s_{new} \leftarrow regularPerturbation(s_{current})$ ▷ Exchange of two operations
15: **end if**
16: $mks_{new}, tds_{new}, flt_{new} \leftarrow calculateValues(s_{new})$
17: **if** $(mks_{new} < mks_{current})$ AND $(tds_{new} < tds_{current})$ AND $(flt_{new} < flt_{current})$ **then**
18: $s_{current} \leftarrow s_{new}$
19: **end if**
20: **end for**
21: **else**
22: $s_{new} \leftarrow regularPerturbation(s_{current})$
23: $mks_{new}, tds_{new}, flt_{new} \leftarrow calculateValues(s_{new})$
24: **end if**
25: **if** $(mks_{new} \neq mks_{current})$ AND $(tds_{new} \neq tds_{current})$ AND $(flt_{new} \neq flt_{current})$ **then**
26: $verifyDominanceCMOSA(T_k, s_{new}, s_{current})$ ▷ See Algorithm 5
27: **end if**
28: **end for**
29: **if** $verifyCaught = TRUE$ **then**
30: **if** $caught(s_{current}, counterTrapped) = TRUE$ **then** ▷ See Algorithm 3
31: $countCaught = countCaught + 1$
32: **if** $countCaught = MAXSTAGNANT$ **then**
33: $verifyCaught \leftarrow FALSE$
34: **end if**
35: **end if**
36: **end if**
37: $L_k \leftarrow \beta \times L_k$
38: $T_k \leftarrow \alpha \times T_k$
39: **end while**
40: **return** $s_{current}$
41: **end procedure**

Algorithm 3 shows the process that is carried out to verify the stagnation mentioned in line 30 of Algorithm 2.

Algorithm 3 Caught

1: **procedure** CAUGHT($s_{current}, counterTrapped$)
2: $\quad isCaught \leftarrow FALSE, timesDominated \leftarrow 0, maxTrapped \leftarrow 10$
3: $\quad timesDominated \leftarrow countTimesDominated(s_{current})$
4: $\quad$ **if** $timesDominated = 0$ **then**
5: $\qquad F \leftarrow s_{current}$
6: $\quad$ **end if**
7: $\quad$ **if** $timesDominated \geq 1$ **then**
8: $\qquad counterTrapped \leftarrow counterTrapped + 1$
9: $\quad$ **end if**
10: $\quad$ **if** $counterTrapped = maxTrapped$ **then**
11: $\qquad isCaught \leftarrow TRUE$
12: $\qquad counterTrapped \leftarrow 0$
13: $\quad$ **end if**
14: $\quad$ **return** $isCaught$
15: **end procedure**

In this Algorithm 3 the current solution ($s_{current}$) and the counter of times it has trapped ($counterTrapped$) are received as input. In line 2 the variables used are initialized. Then the times that the current solution is dominated by at least one solution from the non-dominated front are counted (line 3). If the current solution is non-dominated (line 4) it is stored in the front of non-dominated solutions (line 5). If the current solution is dominated by at least one solution (line 7) then the $counterTrapped$ is incremented (line 8). When $counterTrapped$ equals the maximum number of trapped allowed (line 10), the value of $isCaught$ is set to $TRUE$ (line 11) and the trap counter is reset to zero in line 12.

5.3.2. Chaotic Perturbation

The logistic equation or logistic map is a well-known mathematical application of the biologist Robert May for a simple demographic model [39]. This application tells us the population in the n-th generation based on the size of the previous generation. This value may be found by a popular logistic model mathematically expressed as:

$$x_{n+1} = rx_n(1 - x_n) \tag{25}$$

In Equation (25), the variable x_n takes values ranged between zero and one. This variable represents the fraction of individuals in a specific situation (for instance, into a territory or with a particular feature) in a given instant n. The parameter r is a positive number representing the combined ratio between reproduction and mortality. Even though we are not interested in this paper in demographic or similar problems, we notice the very fast last variable changes. Then it can be taken as a chaotic variable. Thus, we use this variable for performing a chaotic perturbation function, which may help to escape from local optima for our CMOTA and CMOSA algorithms.

The chaotic function used is very sensitive to changes in the initial conditions, and this characteristic is used to generate a perturbation to the solution for escaping from local optima. Then chaos or chaotic perturbation is a process carried out to restart the search from another point in the space of solutions.

Algorithm 4 can be explained in three steps. Firstly, the feasible operations (operations that can be performed without violating any restrictions) are searched (line 4). Secondly, whether there is only one feasible operation (line 5) means that it is the last operation and selected (line 6). When there is more than one feasible operation, a chaotic function is applied to select the operations. In this case, the logistic function is used (lines 8–19), which applies a threshold in the range [0.5 to 1]. Finally, the selected operation is added to the new solution (line 21). This process is applied until all the operations are selected.

Algorithm 4 Chaotic perturbation

1: **procedure** ChaoticPerturbation($s_{current}$)
2: $feasibleTasksNumber \leftarrow 0, r \leftarrow 4, repeat \leftarrow TRUE, X_n \leftarrow 0, X_{n1} \leftarrow 0$
3: **while** $counter < tasks$ **do**
4: $feasibleTasksNumber \leftarrow searchFeasibleTasks()$
5: **if** $feasibleTasksNumber = 1$ **then**
6: $index \leftarrow 0$
7: **else**
8: **while** $repeat = TRUE$ **do**
9: $X_n \leftarrow random(0, 1)$
10: **for** $i \leftarrow 0$ to $feasibleTasksNumber$ **do**
11: $X_{n1} \leftarrow (r \times X_n) \times (1.0 - X_n)$
12: **if** $X_{n1} > 0.5$ **then**
13: $index \leftarrow i$
14: $repeat \leftarrow FALSE$
15: $break$
16: **end if**
17: $X_n \leftarrow X_{n1}$
18: **end for**
19: **end while**
20: **end if**
21: $s_{new} \leftarrow addTask(index)$
22: $counter \leftarrow counter + 1$
23: **end while**
24: **return** s_{new}
25: **end procedure**

5.3.3. Applying Dominance to Select Solutions

In Algorithm 5, the current solution ($s_{current}$) is compared with the new solution (s_{new}) to determine which solution is used to continue the search. In this comparison, there are three cases:

1. If s_{new} dominates $s_{current}$, then s_{new} is used to continue the search (lines 3 to 6).
2. If s_{new} is dominated by $s_{current}$ then the differences of each objective are calculated separately from the two solutions compared to obtain the decreased parameter (Δ) and use it to determine if the s_{new} continues with the search according to the condition in line 12. In this case, $s_{current}$ is added to the non-dominated front (F) and s_{new} replaces $s_{current}$ (lines 13 and 14).
3. If the two solutions are non-dominated by each other, then the current solution $s_{current}$ is added to the non-dominated front (F), and the search continues with s_{new} (lines 18 to 21).

Algorithm 5 Verify dominance CMOSA

procedure VERIFYDOMINANCECMOSA($T_k, s_{new}, s_{current}, mks_{new}, tds_{new}, flt_{new}, mks_{current}, tds_{current}, flt_{current}$)
  $newDominateCurrent \leftarrow FALSE, currentDominateNew \leftarrow FALSE$
  if $s_{new} \prec s_{current}$ **then**
    $s_{current} \leftarrow s_{new}$
    $newDominateCurrent \leftarrow TRUE$
  end if
  if $s_{current} \prec s_{new}$ **then**
    $\Delta_{MKS} \leftarrow mks_{new} - mks_{current}$
    $\Delta_{TDS} \leftarrow tds_{new} - tds_{current}$
    $\Delta_{FLT} \leftarrow flt_{new} - flt_{current}$
    $\Delta \leftarrow \Delta_{MKS} + \Delta_{TDS} + \Delta_{FLT}$
    if $random(0,1) < e^{-\Delta/T_k}$ **then**
      $F \leftarrow s_{current}$
      $s_{current} \leftarrow s_{new}$
    end if
    $currentDominateNew \leftarrow TRUE$
  end if
  if $(newDominateCurrent = FALSE)$ AND $(currentDominateNew = FALSE)$ **then**
    $F \leftarrow s_{current}$
    $s_{current} \leftarrow s_{new}$
  end if
  return $s_{current}$
end procedure

5.4. Chaotic Multi-Objective Threshold Accepting (CMOTA)

In 1990, Dueck et al. proposed the TA algorithm as a general-purpose algorithm for the solution of combinatorial optimization problems [6]. This TA algorithm has a simpler structure than SA, and is very efficient for solving many problems but has never been applied for MOJSSP. The difference between SA and TA is basically in the criteria for accepting bad solutions. TA accepts every new configuration, which is not much worse than the old one. In contrast, SA would accept worse solutions only with small probabilities. An apparent advantage of TA is that it is higher simply because it is not necessary to compute probabilities or to make decisions based on a Boltzmann probability distribution.

Algorithm 6 shows CMOTA algorithm, where we observe that it has the same structure as CMOSA algorithm. These two algorithms have a temperature cycle and, within it, a Metropolis cycle. In these algorithms, a perturbation is applied to the current solution. Then, the dominance of the two solutions is verified to determine which of them is used to continue the searching process (Algorithm 7). Finally, the increment of the variable that controls the iterations of the Metropolis cycle, the reduction of the temperature, and the increment of the counter (line 39) for the number of temperatures are performed.

In Algorithm 7, the dominance of the two solutions is verified to determine which continues with the search. It has the same three cases used in CMOSA (Algorithm 5). The main differences are the following:

- In the beginning, while the temperature counter (*counter*) is less than the value of bound (line 4) T has a value equal to T_k (line 5), which is too large, which implies that at high temperature, the new solution (s_{new}) will often be accepted to continue the search. That is, during the processing of 95% temperatures (parameter $limit = 0.95$, whose value is obtained with Equation (19) in the tuning process), the parameter γ is used to obtain the value T (threshold), and since γ is equal to 1, then it means that T has the value of T_k. For the five percent of the remaining temperatures, γ takes the value of $\gamma_{reduced}$ (0.978). This parameter is tuned experimentally (line 12), and it is established to control the acceptance criterion and make it more restrictive as part of the process.
- CMOTA includes a verification process for accepting bad solution lighting different from CMOSA. To determine if the searching process continues using a dominated solution, CMOTA does not use the Boltzmann criterion to accept it as the current solution. Instead, CMOTA uses a threshold defined as the T parameter value (line 19), which is updated in line 29. In other words, it is no longer necessary to calculate the decrement of the objective functions. This modification makes CMOTA much more

straightforward than CMOSA or any other AMOSA algorithm. Moreover, because the parameter γ is usually very close to one, it is unnecessary to calculate probabilities for the Boltzmann distribution or make a random decision process for bad solutions.

Algorithm 6 Chaotic Multi-Objective Threshold Accepting (CMOTA)

```
procedure CMOTA(T_initial, T_final, α, β, L_k, s_current)
    counter ← 1, MAXSTAGNANT ← 10, counterTrapped ← 0, isCaught ← FALSE
    iterationsLocalSearch ← tasks × timesLS, verifyCaught ← TRUE, countCaught ← 0
    mks_current, tds_current, flt_current ← calculateValues(s_current)    ▷ mks : makespan, tds : tardiness, flt : flowtime
    T_k ← T_initial
    while T_k ≥ T_final do
        for i ← 0 to L_k do
            if isCaught = TRUE then
                isCaught = FALSE
                for k ← 0 to iterationsLocalSearch do
                    if k = 0 then
                        s_new ← chaoticPerturbation(s_current)    ▷ See Algorithm 4
                    else
                        s_new ← regularPerturbation(s_current)    ▷ Exchange of two operations
                    end if
                    mks_new, tds_new, flt_new ← calculateValues(s_new)
                    if (mks_new < mks_current) AND (tds_new < tds_current) AND (flt_new < flt_current) then
                        s_current ← s_new
                    end if
                end for
            else
                s_new ← regularPerturbation(s_current)
                mks_new, tds_new, flt_new ← calculateValues(s_new)
            end if
            if (mks_new ≠ mks_current) AND (tds_new ≠ tds_current) AND (flt_new ≠ flt_current) then
                verifyDominanceCMOTA(counter, T_k, s_new, s_current)    ▷ See Algorithm 7
            end if
        end for
        if verifyCaught = TRUE then
            if caught(s_current, counterTrapped) = TRUE then    ▷ See Algorithm 3
                countCaught = countCaught + 1
                if countCaught = MAXSTAGNANT then
                    verifyCaught ← FALSE
                end if
            end if
        end if
        L_k ← β × L_k
        T_k ← α × T_k
        counter ← counter + 1
    end while
    return s_current
end procedure
```

Algorithm 7 Verify dominance CMOTA

```
procedure VERIFYDOMINANCECMOTA(counter, T_k, s_new, s_current)
    γ ← 1, γ_reduced ← 0.978, setT ← 1, bound ← NumberOfTemperatures × limit
    newDominateCurrent ← FALSE, currentDominateNew ← FALSE
    if counter < bound then
        T ← T_k
    end if
    if (counter = bound) AND (setT = 1) then
        setT ← 0
        T ← T_k
    end if
    if setT = 0 then
        γ ← γ_reduced
    end if
    if s_new ≺ s_current then
        s_current ← s_new
        newDominateCurrent ← TRUE
    end if
    if s_current ≺ s_new then
        if random(0, 1) < T then
            F ← s_current
            s_current ← s_new
        end if
        currentDominateNew ← TRUE
    end if
    if (newDominateCurrent = FALSE) AND (currentDominateNew = FALSE) then
        F ← s_current
        s_current ← s_new
    end if
    T ← T × γ
end procedure
```

6. Main Methodology for CMOSA and CMOTA

Figure 1 shows the main module for each of the two proposed algorithms CMOSA and CMOTA, which may be considered the main processes in any high-level language.

In this main module, the instance to be solved is read, then the tuning process is performed. The due date is calculated, which is an essential element for calculating the tardiness. The set of initial solutions (S) is generated randomly, as follows. First, a collection of feasible operations are determined, then one of them is randomly selected and added to the solution until all the job operations are added.

Once the set of initial solutions has been generated, an algorithm (CMOSA or CMOTA) is applied to improve each initial solution, and the generated solution is stored in a set of final solutions (F). To obtain the set of non-dominated solutions, also called the zero front (f_0) from the set of final solutions, we applied the fast non-dominated Sorting algorithm [29]. To know the quality of the non-dominated set obtained, the MID, Spacing, HV, Spread, IGD, and Coverage metrics are calculated. To perform the calculation of the spread and IGD, the true Pareto front (PF_{true}) is needed. However, for the instances used in this paper, the PF_{true} has not been published for all the instances. For this reason, the calculation was made using an approximate Pareto front (PF_{approx}), which we obtained from the union of the fronts calculated with previous executions of the two algorithms presented here (CMOSA and CMOTA).

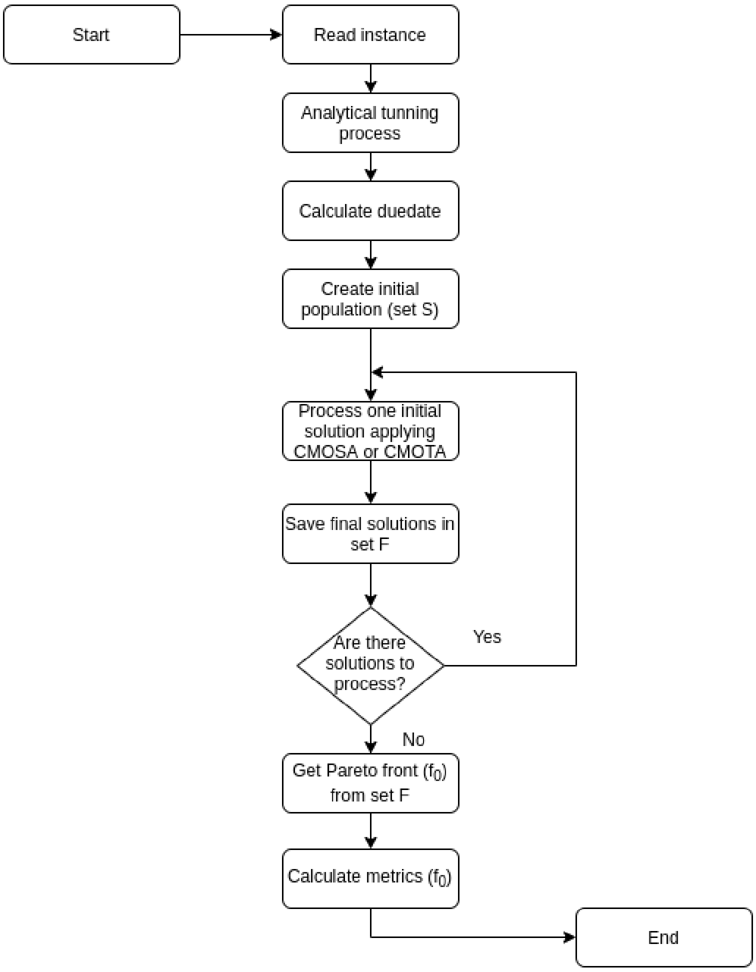

Figure 1. Main module for CMOSA and CMOTA.

6.1. Computational Experimentation

A set of 70 instances of different authors was used to evaluate the performance of the algorithms, including: (1) FT06, FT10, and FT20 proposed by [40]; (2) ORB01 to ORB10 proposed by [41]; (3) LA01 to LA40 proposed by [42]; (4) ABZ5, ABZ6, ABZ7, ABZ8, and ABZ9 proposed by [43]; (5) YN1, YN2, YN3, and YN4 proposed by [44], and (6) TA01, TA11, TA21, TA31, TA41, TA51, TA61, and TA71 proposed by [30].

As already explained, to perform the analytical tuning, some previous executions of the algorithm are necessary. The parameters used for those previous executions are shown in Table 2, and the parameters used in the final experimentation for each instance are shown in Table 3.

Table 2. Tuning parameters for CMOSA/CMOTA.

Number of Executions	Initial Temperature	Final Temperature	Alpha	L_k
50	100	0.1	0.98	100

Table 3. General parameters for CMOSA/CMOTA.

Number of Executions	Initial Solutions	Alpha	Stagnant Number
30	30	0.98	10

The execution of the algorithm was carried out on one of the terminals of the Ehecatl cluster at the TecNM/IT Ciudad Madero, which has the following characteristics:

Intel® Xeon® processor at 2.30 GHz, Memory: 64 GB (4 × 16 GB) ddr4-2133, Linux operating system CentOS, and C language was used for the implementation. We developed CMOSA (https://github.com/DrJuanFraustoSolis/CMOSA-JSSP.git) and CMOTA (https://github.com/DrJuanFraustoSolis/CMOTA-JSSP.git) and we tested the software and using three data sets reported in the paper and taken from the literature.

In the first experiment, the algorithms CMOSA and CMOTA were compared with AMOSA algorithm using the 70 described instances and six performance metrics. In a second experiment, we compared CMOSA and CMOTA with the IMOEA/D algorithm, with the 58 instances used by Zhao [14]. In the second experiment, we used the same MID metric of this publication. The third experiment was based on the 15 instances reported in [8], where the results of the next MOJSSP algorithms are published: SPEA, CMOEA, MOPSO, and MOMARLA. In this publication the authors used two objective functions and two metrics (HV and Coverage); they determined that the best algorithm is MOMARLA followed by MOPSO. We executed CMOSA and CMOTA for the instances of this dataset and we compared our results using the HV metric with those published in [8]. However, a comparison using the coverage metric was impossible because the Pareto fronts of these methods have not been reported [8]. In our case, we show in Appendix A the fronts of non-dominated solutions obtained with 70 instances.

6.2. Results

The average values of 30 runs, for the six metrics obtained by CMOSA and CMOTA for the complete data set of 70 instances are shown in Tables 4 and 5. We observed that CMOSA obtained the best values for MID and IGD metrics. For Spacing and Spread, CMOTA obtained the best results. For the HV metric, both algorithms achieved the same result (0.42). We observed in Table 5 that CMOSA obtained the best coverage result.

A two-tailed Wilcoxon test was performed with a significance level of 5% (last column in Table 4) and this shows that there are no significant differences between the CMOSA and CMOTA except in MID and IGD metrics.

Table 4. Results obtained by the metrics for 70 instances.

Metric	CMOSA	CMOTA	Significant Difference CMOSA-CMOTA
MID	30,680.19 *	31,233.15	Yes
SPACING	28,445.62	28,183.17 *	No
SPREAD	24,969.31	23,401.88 *	No
HV	0.42 *	0.42 *	No
IGD	1666.25 *	1870.94	Yes

* Best result.

Table 5. Results obtained by the coverage metric.

Coverage (CMOSA, CMOTA)	Coverage (CMOTA, CMOSA)
0.854 *	0.063

* Best result.

Table 6 shows the comparison of CMOSA and AMOSA. We observed that CMOSA obtains the best performance in all the metrics evaluated. In addition, the Wilcoxon test indicates that there are significant differences in most of them; thus, CMOSA overtakes AMOSA. We compared CMOTA and AMOSA in Table 7. In this case, CMOTA also obtains the best average results in all the metrics; however, according to the Wilcoxon test, there are significant differences in only two metrics.

Table 6. Comparison among CMOSA with AMOSA.

Metric	CMOSA	AMOSA [17]	Significant Difference CMOSA-AMOSA
MID	30,680.19 *	32,138.19	Yes
SPACING	28,445.62 *	30,129.36	Yes
SPREAD	24,969.31 *	26,625.04	No
HV	0.42 *	0.37	No
IGD	1666.25 *	2209.96	Yes

* Best result.

Table 7. Comparison among CMOTA with AMOSA.

Metric	CMOTA	AMOSA [17]	Significant Difference CMOTA-AMOSA
MID	31,233.15 *	32,138.19	No
SPACING	28,183.17 *	30,129.36	Yes
SPREAD	23,401.88 *	26,625.04	No
HV	0.42 *	0.37	No
IGD	1870.94 *	2209.96	Yes

* Best result.

We compare in Table 8 the CMOSA and CMOTA with the IMOEA/D algorithm using the 58 common instances published in [14] where the MID metric was measured. This table shows the MID average value of this metric for the non-dominated set of solutions of CMOSA and CMOTA. The results showed that CMOSA and CMOTA obtain better performances than IMOEA/D. We notice that both algorithms, CMOSA and CMOTA, achieved smaller MID values than IMOEA/D, which indicates that the Pareto fronts of our algorithms are closer to the reference point (0,0,0). The Wilcoxon test confirms that CMOSA and CMOTA surpassed the IMOEA/D.

Table 8. CMOSA, CMOTA, and IMOEA/D results obtained using MID metric.

CMOSA	CMOTA	IMOEA/D [14]	Significant Difference CMOSA-IMOEA/D	Significant Difference CMOTA-IMOEA/D
15,729.65 *	16,567.07	18,727.04	Yes	Yes

* Best result.

The results of CMOSA and CMOTA were compared with the SPEA, CMOEA, MOPSO, and MOMARLA algorithms [8]. In the last reference, only two objective functions were reported, the makespan and total tardiness. The experimentation was carried out with 15 instances and the average HV values were calculated to perform the analysis of the results, which are shown in Table 9. We notice that MOMARLA surpassed SPEA, CMOEA, and MOPSO. We can observe that CMOSA obtained a better performance than MOMARLA and the other algorithms. Comparing CMOTA and MOMARLA, we notice that both algorithms obtained the same HV average results.

Table 9. Comparison among SPEA, CMOEA, MOPSO, CMOSA, CMOTA, and MOMARLA using HV.

	Instance	SPEA [8]	CMOEA [8]	MOPSO [8]	MOMARLA [8]	CMOSA	CMOTA
1	FT06	0.07	0.07	0.50	0.65	0.64	0.75 *
2	FT10	0.17	0.26	0.87	0.96	0.71	0.69
3	FT20	0.20	0.20	0.21	0.25	0.57 *	0.77 *
4	ABZ5	0.34	0.33	0.36	0.40	0.85 *	0.56 *
5	ABZ6	0.22	0.36	0.31	0.42	0.60 *	0.81 *
6	ABZ7	0.51	0.45	1.00	1.00	0.79	0.51
7	ABZ8	0.88	0.36	0.99	0.99	0.69	0.66
8	LA26	0.33	0.39	0.47	0.47	0.91 *	0.70 *
9	LA27	0.58	0.56	0.41	0.60	0.71 *	0.93 *
10	LA28	0.48	0.42	0.48	0.54	0.92 *	0.44
11	ORB01	0.62	0.74	0.59	0.80	0.87 *	0.63
12	ORB02	0.20	0.04	0.30	0.53	0.88 *	0.77 *
13	ORB03	0.69	0.31	0.85	0.86	0.76	0.80
14	ORB04	0.63	0.28	0.52	0.79	0.76	0.81 *
15	ORB05	0.00	0.023	0.22	0.90	0.74	0.32
	Mean HV	0.39	0.32	0.54	0.68	0.76 *	0.68

* Best result.

6.3. CMOSA-CMOTA Complexity and Run Time Results

In this section, we present the complexity of the algorithms analyzed in this paper. The algorithms' complexity is presented in Table 10, and it was obtained directly when it was explicitly published or determined from the algorithms' pseudocodes. In this table, M is the number of objectives, Γ is the population size, T is the neighborhood size, n is the number of iterations (temperatures for AMOSA, CMOSA, and CMOTA), and p is the problem size. The latter is equal to jm where j and m are the number of jobs and machines, respectively. Because the algorithms with the best quality metrics are CMOSA, CMOTA MOMARLA, and MOPSO, their complexity is compared in this section.

It is well known that the complexity of classical SA is $O(p^2 \, log \, p)$ [45]. However, we notice from Table 10 that CMOSA, and CMOTA have a different complexity even though they are based on SA. This is because these new algorithms applied a different chaotic perturbation and another local search (see Algorithms 2 and 6 in lines 10–20).

The temporal function of MOMARLA, CMOSA, and CMOTA belong to $O(Mnp)$. For MOMARLA, n is the number of iterations, a variable used at the beginning of this algorithm. On the other hand, for CMOSA and CMOTA, n is the number of temperatures used in the algorithm, also at its beginning; in any case, the difference will be only a constant.

We note that AMOSA and MOPSO have a similar complexity class expression, that is $O(n\Gamma^2)$ and $O(M\Gamma^2)$ respectively. However, MOPSO overtakes AMOSA because M is in general lower than n. We observe that CMOSA, CMOTA and MOMARLA belong to $O(Mnp)$ class complexity, while MOPSO belongs to $O(M\Gamma^2)$ [46]. Thus, the relation between them is np/Γ^2 which in general is lower than one. Thus CMOSA, CMOTA and MOMARLA have a lower complexity than MOPSO. Moreover, CMOSA, CMOTA, and MOMARLA have better HV metric quality as is shown in Table 9.

In the next paragraph, we present a comparative analysis of the execution time of the algorithms implemented in this paper.

Table 10. Complexity of the algorithms.

AMOSA	IMOEA/D	SPEA	MOPSO	MOMARLA	CMOSA	CMOTA
$O(n\Gamma^2)$	$O(M\Gamma T)$	$O(M\Gamma)$	$O(M\Gamma^2)$	$O(Mnp)$	$O(Mnp)$	$O(Mnp)$

In Table 11 we show the execution time, expressed in seconds, for the three algorithms (CMOSA, CMOTA, and AMOSA) implemented in this paper for three data sets (70, 58, and 15 instances). In all these cases, we emphasize that the AMOSA algorithm was the base to design the other two algorithms. In fact, all of them have the same structure except that CMOSA and CMOTA apply chaotic perturbations when they detect a possible stagnation. Thus, all of them have similar complexity measures for the worst-case. Table 11 shows the percentage of time saved by these two algorithms concerning AMOSA. For these datasets, we measured that AMOSA saved 2.1, 19.87, and 42.48 percent of the AMOSA run time; on the other hand, these figures of CMOTA versus AMOSA are 55, 68.89, and 46.73 percent. Thus, both of our proposed algorithms CMOSA and CMOTA are significantly more efficient than AMOSA. Unfortunately, we do not have the tools to compare these algorithms versus the other algorithms' execution time in Table 1. Nevertheless, we made the quality comparisons by using the metrics previously published.

Table 11. Runtimes for CMOSA, CMOTA and AMOSA.

Algorithm	CMOSA	CMOTA	AMOSA [17]
Data set of 70 instances			
Average execution time	495.22	229.42 *	505.84
% time saved vs AMOSA	2.1	55 *	0
Data set of 58 instances			
Average execution time	111.68	41.97 *	139.39
% time saved vs AMOSA	19.87	69.89 *	0
Data set of 15 instances			
Average execution time	81.24	75.24 *	141.25
% time saved vs AMOSA	42.48	46.73 *	0

* Best result.

7. Conclusions

This paper presents two multi-objective algorithms for JSSP, named CMOSA and CMOTA, with three objectives and six metrics. The objective functions for these algorithms are makespan, total tardiness, and total flow time. Regarding the results from the comparison of CMOSA and CMOTA with AMOSA, we observe that both algorithms obtained a well-distributed Pareto front, closest to the origin, and closest to the approximate Pareto front as was indicated by Spacing, MID, and IGD metrics, respectively. Thus, using these five metrics, we found that CMOSA and CMOTA surpassed the AMOSA algorithm. Regarding the volume covered by the front calculated by the HV metric, it was observed that both algorithms, CMOSA and CMOTA, have the same performance; however, CMOSA has a higher convergence than CMOTA. In addition, the proposed algorithms surpass IMOEA/D when MID metric was used. Moreover, we use the HV to compare the proposed algorithms with SPEA, CMOEA, MOPSO, and MOMARLA. We found that CMOSA outperforms these algorithms, followed by CMOTA, MOMARLA, and MOPSO.

We observe that CMOSA and CMOTA have similar complexity as the best algorithms in the literature. In addition, we show that CMOSA and CMOTA surpass AMOSA when we compare them using execution time for three data sets. We found CMOTA is, on average, 50 percent faster than AMOSA and CMOSA. Finally, we conclude that CMOSA and CMOTA have similar temporal complexity than the best literature algorithms, and the quality metrics show that the proposed algorithms outperform them.

Author Contributions: Conceptualization: J.F.-S., L.H.-R., G.C.-V.; Methodology: J.F.-S., L.H.-R., G.C.-V., J.J.G.-B.; Investigation: J.F.-S., L.H.-R., G.C.-V., J.J.G.-B.; Software: J.F.-S., L.H.-R., G.C.-V., J.J.G.-B.; Formal Analysis: J.F.-S., G.C.-V.; Writing original draft: J.F.-S., L.H.-R., G.C.-V.; Writing review and editing: J.F.-S., J.J.G.-B., J.P.S.-H. All authors have read and agreed to the published version of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to express their gratitude to CONACYT and TecNM/IT Ciudad Madero. In addition, the authors acknowledge the support from Laboratorio Nacional de Tecnologías de la Información (LaNTI) for the access to the cluster.

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

Appendix A. Non-Dominated Front Obtained

The non-dominated solutions obtained by CMOSA algorithm for the 70 instances used are shown in Tables A1–A6, and the non-dominated solutions obtained by CMOTA algorithm for the same instances are shown in Tables A7–A12. In these tables, MKS is the makespan, TDS is the total tardiness and FLT is the total flow time. For each instance, the best value for each objective function is highlighted with an asterisk (*) and in bold type.

Table A1. Non-dominated front obtained by CMOSA for the JSSP instances proposed by [40].

	FT06			FT10			FT20		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**55** *	30.0	305	**993** *	1768.5	9234	**1224** *	8960.0	16614
2	55	38.0	301	994	1609.0	9121	1227	8809.0	16375
3	56	37.0	304	1004	1495.0	9062	1229	8793.0	16359
4	56	29.0	308	1006	1083.0	8584	1235	8774.0	16340
5	57	23.5	305	1036	1053.0	**8406** *	1243	**8455.5** *	**16119** *
6	57	27.0	297	1037	**1009.0** *	8437			
7	57	26.0	298						
8	58	9.5	280						
9	60	11.0	**279** *						
10	62	8.5	285						
11	69	**8.0** *	291						

Table A2. Non-dominated front obtained by CMOSA for the JSSP instances proposed by [41].

	ORB1			ORB2			ORB3			ORB4			ORB5		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1142** *	1539.0	9245	**925** *	767.5	8339	**1104** *	1874.0	9448	**1063** *	1186.0	9175	**966** *	1192.5	8279
2	1143	1517.0	9223	927	781.5	8285	1111	1548.0	9392	1073	1108.5	9270	971	1180.5	8296
3	1144	1522.0	9135	931	722.5	8160	1112	1816.0	9318	1078	1059.5	9128	975	859.5	7648
4	1150	1381.5	9219	951	542.5	8056	1123	1462.0	9306	1107	917.5	9234	978	752.5	8016
5	1161	**1355.5** *	9469	958	**331.0** *	7742	1127	1806.0	9288	1111	978.0	9199	980	758.5	8011
6	1172	1508.0	9214	958	339.0	**7730** *	1162	1579.0	9200	1134	944.5	9221	984	708.5	7961
7	1174	1521.0	**9134** *				1164	1562.0	9183	1140	795.5	9111	984	706.5	7970
8							1180	1492.5	8984	1156	843.5	9083	998	822.0	7784
9							1187	**1475.5** *	**8967** *	1200	**733.5** *	9049	1001	746.5	7869
10										1230	919.0	8969	1001	834.0	**7620** *
11										1232	983.5	8813	1013	**689.0** *	7765
12										1277	995.5	**8735** *	1017	795.0	7713
13													1032	798.0	7659
14													1049	771.0	7678

Table A2. *Cont.*

	ORB6			ORB7			ORB8			ORB9			ORB10		
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1097** *	1318.0	9573	**423** *	207.5	3663	**963** *	1804.0	8439	**987** *	1193.5	8912	**991** *	835.0	8482
2	1100	1199.5	9505	424	167.0	3731	968	1412.5	8204	988	1362.5	8860	993	843.0	8465
3	1100	1267.5	9434	431	**161.0** *	3643	970	1387.0	8215	993	1220.0	8898	1020	798.5	8785
4	1105	1225.0	9434	439	295.0	3620	988	1514.5	8099	996	1072.5	8844	1029	742.5	8691
5	1105	1227.0	9412	449	207.5	3625	997	1587.0	8078	1006	1002.0	8538	1043	608.5	8659
6	1110	1255.0	9409	453	230.5	3616	1001	1239.0	7912	1019	1017.5	8523	1044	**493.5** *	8522
7	1113	1220.5	9452	455	204.5	3636	1044	**1120.0** *	**7617** *	1035	1100.5	8493	1072	774.5	**8455** *
8	1114	1078.5	9287	459	213.0	3577				1039	1043.5	8430			
9	1141	1153.0	**9109** *	461	216.0	3509				1048	**887.0** *	**8348** *			
10	1171	1097.0	9194	461	203.0	3545									
11	1191	1018.5	9145	461	186.5	3572									
12	1233	**988.0** *	9225	466	202.5	3547									
13				466	171.0	3561									
14				470	184.5	**3504** *									

Table A3. Non-dominated front obtained by CMOSA for the JSSP instances proposed by [42].

	LA01			LA02			LA03			LA04			LA05		
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**666** *	1194.0	5436	**655** *	1207.0	5123	**615** *	1492.5	5000	**590** *	1252.0	4900	**593** *	1159.5	4451
2	666	1237.5	5362	656	1161.0	5077	622	1400.5	4896	595	1235.0	4948	593	1088.0	4455
3	667	1382.5	5357	665	1222.0	4994	626	1484.5	4881	598	1250.0	4857	594	1053.0	4399
4	668	1068.5	5328	665	1203.0	5050	627	1467.0	4889	598	1226.5	4910	610	1099.5	4386
5	668	1074.0	5309	671	1042.0	4904	628	1343.5	4866	599	1167.0	4915	615	1129.5	**4351** *
6	670	1269.5	5300	673	1094.5	4879	630	1357.5	4803	603	1154.5	4895	631	**999.5** *	4371
7	672	1152.5	5260	681	938.5	4799	630	1339.5	4850	605	1089.0	4737	648	1036.0	4359
8	688	1145.5	5247	695	927.5	4864	633	1226.5	4750	614	1034.0	4782	659	1032.0	4355
9	700	1120.5	5297	695	930.5	4796	638	1183.0	4649	615	1047.5	4756			
10	706	1081.5	5241	696	910.5	4837	641	1178.5	4713	618	1042.5	4705			
11	706	1179.0	5225	714	997.5	4776	646	1173.0	4718	622	1038.5	4705			
12	713	1065.5	5203	715	936.5	4720	655	1088.5	4482	629	1006.0	4710			
13	718	1025.5	5235	736	925.0	4812	662	1062.0	4595	629	1020.5	4695			
14	727	1056.5	5138	741	993.0	**4716** *	662	1081.5	4591	631	982.5	4697			
15	734	1046.0	5184	771	**909.5** *	4786	668	1015.0	4522	637	981.0	4576			
16	743	1089.0	5101				669	981.5	4523	638	961.5	4667			
17	751	**951.0** *	5115				683	979.5	4516	640	962.0	4566			
18	825	1098.0	**5099** *				688	1087.5	4481	643	930.0	**4525** *			
19							698	1055.0	4504	648	927.0	4531			
20							741	955.5	4382	650	895.5	4558			
21							744	891.0	4375	655	908.0	4537			
22							744	914.0	4372	663	**888.5** *	4551			
23							750	896.5	**4323** *	663	906.0	4543			
24							757	**867.0** *	4325						

Table A3. *Cont.*

		LA06			**LA07**			**LA08**			**LA09**			**LA10**	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**926 ***	4185.5	10,142	**890 ***	4006.5	9554	**863 ***	3717.5	9455	**951 ***	3925.0	10,297	**958 ***	4439.5	10,441
2	927	4183.0	10,171	890	4044.0	9496	863	3792.5	9424	951	3916.5	10,311	969	4476.5	10,437
3	929	4062.0	10,050	894	3974.5	9522	865	3723.5	9387	954	**3908.0 ***	10,280	971	4313.0	10,343
4	931	4122.0	10,041	896	3646.5	9264	870	3685.5	9349	974	3944.5	**10,195 ***	976	4298.0	10,328
5	938	3911.0	9870	904	3684.0	9248	871	3649.5	9284				982	4121.0	10,151
6	940	**3827.0 ***	**9786 ***	906	3615.0	9219	876	3602.5	9340				1052	**4083.0 ***	**10,113 ***
7				910	3652.0	9216	885	3598.5	9309						
8				967	**3595.0 ***	**9199 ***	895	3596.0	9266						
9							896	**3410.5 ***	**9045 ***						

		LA11			**LA12**			**LA13**			**LA14**			**LA15**	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1222 ***	9157.5	17,184	**1039 ***	7218.0	14,229	**1150 ***	8436.5	16,208	**1292 ***	10,017.0	18,036	**1207 ***	9447.5	17,581
2	1225	8947.5	16,853	1041	7203.0	14167	1153	8333.5	16,105	1299	9986.0	18,005	1208	9249.5	17,383
3	1241	8879.5	16,785	1043	7198.0	14196	1154	8310.5	16,079	1328	9992.5	17,990	1213	9175.0	17,314
4	1242	8862.5	16,768	1049	7164.0	14162	1155	8247.5	15,953	1352	**9810.5 ***	17,808	1220	9149.0	17,284
5	1243	8860.5	16,766	1050	7126.0	14124	1161	8175.0	15,954	1352	9867.0	**17,797 ***	1229	9014.0	17,149
6	1256	8811.5	16,798	1134	**7114.0 ***	**14,112 ***	1162	8210.5	15,916				1232	9013.0	17,148
7	1257	8725.5	16,712				1182	8057.0	15,836				1234	8991.0	17,126
8	1258	8765.5	16,671				1183	8013.0	15,792				1251	8915.5	17,062
9	1265	**8650.5 ***	**16,637 ***				1184	7994.0	15,773				1271	8947.5	17,040
10							1185	7989.0	15,768				1273	8703.5	16871
11							1189	**7978.0 ***	**15,757 ***				1281	8651.5	16,819
12													1283	8638.5	16,802
13													1289	8603.5	16,767
14													1297	**8601.5 ***	**16,765 ***

		LA16			**LA17**			**LA18**			**LA19**			**LA20**	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**968 ***	983.5	8777	**796 ***	799.0	7502	**865 ***	488.0	7765	**884 ***	538.0	7950	**934 ***	665.5	8354
2	982	904.0	8754	796	784.0	7509	866	468.5	7743	889	288.0	7945	939	599.5	8409
3	988	898.5	8608	810	855.0	7492	868	439.5	7853	891	495.0	7821	948	631.5	8393
4	992	882.0	8752	811	783.0	7555	873	419.5	7687	900	406.0	7916	957	542.0	8423
5	994	816.5	8669	813	702.0	7458	878	396.5	7755	905	279.0	7846	957	556.0	8302
6	1000	873.0	8570	813	745.0	7450	882	404.5	7732	935	327.0	7730	964	658.0	8232
7	1003	900.0	8565	816	693.0	7458	883	429.5	7648	953	335.5	7726	966	403.0	8032
8	1003	908.0	8545	820	630.0	7395	893	411.0	7671	953	**259.5 ***	7806	967	408.0	8028
9	1003	942.0	8474	823	670.5	7334	923	394.5	7802	979	304.5	**7673 ***	971	408.0	8001
10	1008	493.0	8205	824	633.5	7240	927	368.5	7885				972	419.0	7975
11	1016	553.5	8063	831	623.5	7321	928	351.5	7882				1009	390.5	8094
12	1040	459.5	8232	833	625.5	7320	939	353.0	7691				1067	422.0	7927
13	1050	352.0	7997	835	717.5	7203	939	300.5	7860				1084	424.0	**7908 ***
14	1066	345.5	8285	836	596.5	7291	940	345.0	7827				1100	383.5	8292
15	1071	341.5	8068	836	611.5	7284	945	332.5	7845				1115	382.5	8065
16	1073	401.0	7980	840	597.0	7267	946	305.0	7629				1142	335.5	7915
17	1095	**326.5 ***	**7908 ***	840	612.0	7260	952	**267.0 ***	7778				1142	334.0	7998
18				842	612.0	7194	978	476.0	7614				1148	**262.5 ***	8205
19				849	522.0	7208	982	455.0	**7519 ***				1168	302.5	8204
20				849	521.5	7232	984	439.0	7626						

Table A3. *Cont.*

21				864	531.0	7135	998	361.5	7603						
22				864	530.5	7159									
23				864	521.5	7169									
24				899	535.0	7114									
25				914	509.0	7034									
26				927	**470.0** *	7098									
27				931	475.0	**7000** *									

		LA21			LA22			LA23			LA24			LA25	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1124** *	3229.5	15,030	**1013** *	2968.5	13,774	**1077** *	2292.0	14,222	**1000** *	2145.5	**13230** *	**1071** *	3161.0	14,387
2	1124	3233.5	15,002	1018	2916.5	13,722	1078	2253.5	14,198	1008	2137.5	13,474	1072	3060.0	14,275
3	1127	3180.5	14,883	1020	2906.5	13,712	1078	2249.5	14,238	1008	2120.5	13,606	1089	3002.0	14,096
4	1128	3137.5	14,868	1034	2738.5	13,552	1080	2173.5	14,152	1077	2010.5	13,458	1100	2756.5	13,951
5	1129	3015.5	14,718	1037	2660.0	13,638	1091	2231.5	14,149	1079	1981.5	13,390	1104	2764.5	13,940
6	1137	2998.5	14,400	1038	2774.5	13,548	1095	2243.5	14,147	1088	**19,76.5** *	13,385	1118	2721.0	13,962
7	1141	2892.5	14,636	1039	2648.0	13,611	1097	2071.0	14,011				1118	2768.0	13,938
8	1144	2821.5	14,565	1045	2811.0	13,528	1102	**1939.0** *	**13,867** *				1121	2802.5	13,829
9	1146	2939.0	14,346	1047	2696.5	13,510							1123	2618.5	13,658
10	1150	2543.0	14,344	1050	2614.5	13,445							1131	2584.5	13,845
11	1150	2639.5	14,316	1068	2565.5	13,396							1134	2536.5	13,577
12	1157	2557.5	14,247	1076	2544.5	13,375							1134	2529.0	13,770
13	1158	2545.5	14,222	1082	2462.5	13,253							1154	2517.5	13,535
14	1164	2511.5	14,188	1087	2392.5	13,169							1159	2457.0	13,654
15	1179	2393.5	14,204	1099	**2332.5** *	**13,109** *							1160	2451.5	13,666
16	1182	2331.5	14,165										1173	2530.0	13,470
17	1182	2355.5	14,153										1175	2445.0	13,385
18	1183	2454.5	14,131										1187	2435.0	13,481
19	1227	2328.0	14,238										1189	**2315.0** *	**13,255** *
20	1247	**2225.0** *	14,161												
21	1258	2561.5	13,967												
22	1272	2527.5	13,963												
23	1285	2465.5	**13,871** *												
24	1290	2305.0	14,103												

		LA26			LA27			LA28			LA29			LA30	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1281** *	6921.0	22,576	**1332** *	6555.0	22,803	**1318** *	7579.0	23,547	**1293** *	7971.5	22,802	**1434** *	9177.0	25,172
2	1282	6811.0	22,466	1334	6495.0	22,743	1321	7403.0	23,426	1294	7963.5	22,786	1437	8132.0	24,056
3	1304	6708.5	22,434	1340	6399.0	22,647	1329	6603.0	22,626	1317	7799.5	22,693	1445	8064.0	23,991
4	1323	6643.5	22,416	1346	6280.0	22,528	1362	6683.5	22,578	1319	7796.5	22,690	1448	7996.0	**23,923** *
5	1325	6629.5	22,402	1358	**6228.0** *	**22,476** *	1367	6552.0	22,575	1327	7770.5	22,664	1540	**7980.0** *	24,000
6	1328	6741.5	22,254				1378	6469.0	22,454	1333	7738.5	22,632			
7	1329	6560.5	22,333				1385	6465.0	22,389	1334	7711.5	22,605			
8	1338	6616.5	22,129				1393	6480.5	22,360	1339	7507.5	22,314			
9	1340	6510.5	22,276				1413	6443.0	22,320	1340	7446.5	22,253			
10	1377	**6307.0** *	**21,940** *				1416	6439.0	22,316	1368	7411.5	22,218			
11							1454	6429.0	22,298	1375	7398.5	22,289			
12							1476	6239.0	22,013	1376	7464.5	22182			
13							1477	**6141.0** *	**21,915** *	1376	7374.5	22,268			
14										1379	7018.5	21,912			
15										1389	**7011.5** *	**21,905** *			

Table A3. *Cont.*

	LA31			LA32			LA33			LA34			LA35		
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1784 ***	20,830.5	43,617	**1850 ***	20,861.5	45715	**1719 ***	20,933.5	43,387	**1743 ***	22,605.5	45,617	**1898 ***	24,225.5	47,233
2	1794	20,718.5	43,505	1867	20,860.5	45,714	1721	18,798.5	41,252	1747	21,475.5	44,487	1899	23,434.5	46,652
3	1796	20,390.5	43,177	1871	20,686.5	45,540	1723	18,528.5	40,982	1755	21,271.5	44,283	1900	22,784.5	46,012
4	1797	20,066.5	42,842	1881	20,563.5	45,417	1725	18,137.5	40,591	1756	21,211.5	44,223	1901	22,724.5	45,952
5	1798	20,009.5	42785	1889	20,059.5	44,913	1738	**18,109.5 ***	**40,563 ***	1759	21041.5	44,037	1903	22,684.5	45,912
6	1800	**19,919.5 ***	**42,695 ***	1900	**20,049.5 ***	**44,903 ***				1771	20,916.0	43,916	1920	22,481.5	45,709
7										1774	20,787.0	43,787	1947	22,677.0	45,695
8										1781	20,736.0	43,736	1950	22,442.5	45,670
9										1791	20,693.5	43,705	1953	22,454.0	45,665
10										1801	20,505.5	43,517	1958	22,327.5	45,555
11										1837	20,476.5	43,488	2018	**22,311.5 ***	**45,539 ***
12										1839	20,356.5	43,368			
13										1840	20,305.5	43,317			
14										1843	20,298.5	43,310			
15										1850	20,072.5	43,084			
16										1906	**19,880.5 ***	**42,892 ***			

	LA36			LA37			LA38			LA39			LA40		
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1453 ***	3131.0	20,575	**1569 ***	3065.0	21,444	**1400 ***	1586.0	18,171	**1444 ***	2371.0	19,447	**1436 ***	2617.5	19,260
2	1471	3030.5	20,309	1571	3077.0	21,436	1419	**1578.5 ***	18,200	1452	2056.0	19,215	1443	2017.0	18,689
3	1474	2834.5	20,125	1574	3043.0	21,402	1421	2057.5	18,119	1498	1770.5	18,662	1450	1806.0	18,391
4	1475	2936.5	20,085	1574	3025.0	21,404	1439	2092.5	18,067	1499	1731.5	18,607	1458	1719.0	18,303
5	1476	2847.5	20,094	1580	3009.0	21,301	1468	1753.5	18,103	1504	1473.5	18,404	1471	**1433.5 ***	18,431
6	1476	2949.5	20,054	1584	3002.0	21,294	1473	1736.5	18,086	1621	1422.5	18,579	1495	1549.5	**18,287 ***
7	1487	2633.5	19,889	1590	2331.5	20,755	1496	1744.5	**18,044 ***	1817	**1902.0 ***	**18,191 ***			
8	1498	2474.5	19,694	1593	2289.5	20,748									
9	1505	2492.5	19,675	1608	2247.5	20,585									
10	1521	2604.0	19,671	1614	2384.0	20,153									
11	1521	2379.0	19,840	1614	2414.0	20,101									
12	1529	2459.5	19,679	1618	2374.0	20,143									
13	1530	2420.0	19,668	1621	2418.0	**20,077 ***									
14	1534	2335.5	19,812	1649	2234.5	20,600									
15	1534	2472.5	19,650	1650	2237.5	20,587									
16	1548	2278.5	19,755	1650	2241.5	20,557									
17	1563	**2015.5 ***	19,237	1700	2222.5	20,453									
18	1573	2532.5	**19,231 ***	1700	2205.0	20,517									
19				1707	2187.5	20,418									
20				1781	2012.0	20,554									
21				1781	1964.5	20,634									
22				1790	**1835.5 ***	20,309									

Table A4. Non-dominated front obtained by CMOSA for the JSSP instances proposed by [43].

	ABZ5			ABZ6			ABZ7			ABZ8			ABZ9		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1250** *	145.0	11,006	**967** *	324.0	8453	**746** *	2420.0	13,274	**763** *	23,17.0	13,696	**805** *	3296.5	14,426
2	1250	**134.0** *	11,025	974	256.5	8524	753	2403.0	13,257	763	2332.0	13,688	807	3127.0	14,287
3	1252	139.0	10,998	974	251.5	8550	793	**2305.0** *	**13,137** *	773	2336.0	13,675	808	2941.0	14,094
4	1289	141.0	10,984	979	204.0	8464				773	2326.0	13,688	822	2846.0	13,820
5	1289	142.0	**10,946** *	997	258.5	8357				775	2294.0	13,633	833	2770.0	13,840
6				999	202.0	8553				779	**2236.5** *	**13,591** *	842	2733.5	13,888
7				1001	172.0	8484							843	2740.5	13,845
8				1009	164.0	8589							845	2727.5	13,832
9				1016	164.5	8532							846	2706.5	13,811
10				1018	134.0	8692							847	2696.5	13,801
11				1019	126.0	**8275** *							885	2806.0	13,800
12				1074	35.5	8583							886	2737.0	13,762
13				1077	36.5	8525							889	2726.0	13,720
14				1077	49.5	8459							896	2708.5	13,703
15				1080	25.5	8550							897	**2684.5** *	**13,679** *
16				1082	29.5	8488									
17				1082	40.5	8472									
18				1085	**1.5** *	8423									

Table A5. Non-dominated front obtained by CMOSA for the JSSP instances proposed by [44].

	YN01			YN02			YN03			YN04		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1103** *	2485.0	19,819	**1133** *	2178.0	19,429	**1083** *	2025.5	19,346	**1210** *	2864.5	20,633
2	1105	2442.0	19,776	1137	2205.0	19,424	1084	2015.5	19,336	1221	**2814.0** *	20,552
3	1105	2465.5	19,753	1140	2050.0	19,299	1084	2012.5	19,337	1297	2915.5	20,525
4	1106	2418.5	19,706	1140	2067.0	19,286	1089	2003.5	19,328	1300	2910.5	**20,520** *
5	1106	2395.0	19,729	1148	2059.0	19,278	1090	**1987.5** *	19,308			
6	1108	1901.0	19,129	1150	**2023.0** *	**19,276** *	1138	2179.5	19,219			
7	1111	1859.0	19,068				1203	2157.5	**18,751** *			
8	1117	1867.5	**19,013** *									
9	1126	**1756.5** *	19,265									
10	1131	1772.5	19,247									

Table A6. Non-dominated front obtained by CMOSA for the JSSP instances proposed by [30].

	TA01			TA11			TA21			TA31		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1412** *	1821.5	18,716	**1603** *	6409.5	27,903	**2048** *	7261.5	37,039	**2083** *	20,557.0	54,457
2	1412	16,41.5	18,749	1607	6365.5	27,859	2050	6184.5	36,322	2091	20,504.0	54,404
3	1414	1809.5	18,704	1619	6051.5	27,722	2051	6184.5	36,290	2096	20,448.0	54,348
4	1433	1753.5	**18,648** *	1750	6387.0	27,635	2074	6023.5	36,129	2097	20,112.0	54,012
5	1443	1733.5	18,739	1753	6307.0	**27,555** *	2078	6017.5	36,123	2099	20,099.0	53,999
6	1448	**1625.0** *	18,765	1766	6293.0	27,572	2091	6031.0	36,050	2106	19,879.0	53,779
7				1859	**6088.0** *	27,679	2274	**5393.0** *	**35,462** *	2109	19,860.0	53,760
8										2119	19,857.0	53,757
9										2121	19,802.0	53,702
10										2125	19,782.0	53,682
11										2132	18,670.5	52,157
12										2139	**18,657.5** *	**52,144** *

Table A6. *Cont.*

	TA41			TA51			TA61			TA71		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**2530** *	18,610.5	65,529	**3121** *	77,760.0	134,637	**3437** *	71,924.0	148,370	**6050** *	368,519.5	519,856
2	2553	18,589.5	65,508	3124	74,125.0	131,002	3445	71,162.0	147,608	6063	368,491.5	519,828
3	2731	18,298.0	65,157	3125	74,113.0	130,990	3561	70,685.0	147,131	6097	367,933.5	519,270
4	2733	18,257.0	65,116	3127	74,028.0	130,905	3567	**70,550.0** *	**146,996** *	6098	367,927.5	51,9264
5	2736	18,228.0	65,087	3134	72,636.0	129,513				6129	366,149.5	51,7486
6	2743	18,197.0	65,056	3186	72,624.0	129,501				6165	365,118.5	516,455
7	2832	181,28.5	65,047	3188	71,884.0	128,761				6166	365,116.5	516,453
8	2949	**17,853.5** *	**64,772** *	3189	71,849.0	128,726				6168	365,090.5	516,427
9				3202	70,643.0	127,520				6215	**361,891.5** *	**513,228** *
10				3204	**70,623.0** *	**127,500** *						

Table A7. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [40].

	FT06			FT10			FT20		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**55** *	30.0	305	**1021** *	1759.5	9407	**1234** *	9571.0	17,132
2	55	38.0	301	1029	1721.0	9122	1240	8914.5	16,578
3	56	29.0	308	1063	1711.0	9358	1243	8934.0	16,526
4	57	23.5	305	1065	1697.0	9280	1249	8898.5	16,562
5	57	26.0	298	1067	1562.5	9226	1258	8959.5	16,480
6	57	27.0	297	1088	1650.5	**8859** *	1259	8930.5	16451
7	58	9.5	280	1089	1614.5	9031	1270	8831.5	16,352
8	60	**8.5** *	**276** *	1091	1619.5	9018	1277	8782.5	16,303
9				1109	1468.0	9046	1327	8768.0	16,365
10				1125	1459.0	8890	1351	8768.5	**16,289** *
11				1146	**1361.0** *	9003	1359	**8738.0** *	16,335

Table A8. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [41].

	ORB1			ORB2			ORB3			ORB4			ORB5		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1180** *	1853.0	9764	**964** *	985.5	8421	**1124** *	2307.5	10,157	**1094** *	1727.5	9897	**945** *	1006.0	8032
2	1190	1714.5	9619	983	971.5	8672	1134	1901.0	9579	1104	1720.5	10,062	980	975.0	7992
3	1192	1721.5	9585	985	913.5	8601	1208	1842.5	9770	1109	1695.5	10,117	994	**747.0** *	7966
4	1237	1787.5	9440	986	975.5	8593	1212	1795.5	9721	1111	1600.5	9865	999	751.0	7950
5	1238	1714.5	9616	987	1009.0	8347	1217	1829.5	9698	1118	1507.0	9818	1053	979.5	**7944** *
6	1249	1799.5	9423	988	980.0	8303	1218	1791.5	9717	1130	1626.0	9704			
7	1253	1771.5	9428	991	857.5	8545	1219	1875.0	9531	1132	1588.5	9768			
8	1255	1582.0	9459	996	918.0	8427	1240	**1516.5** *	**9349** *	1133	1595.5	9760			
9	1261	1581.0	9387	1011	842.0	8630				1138	1548.5	9713			
10	1336	1415.5	9303	1015	854.5	8526				1143	1487.0	9798			
11	1339	**1372.5** *	**9260** *	1020	625.5	8251				1153	1626.0	9674			
12				1047	**625.0** *	8288				1155	1472.5	9645			
13				1081	753.0	**8059** *				1165	1452.5	9625			
14				1209	721.5	8224				1165	1440.0	9645			
15										1166	1428.0	9633			
16										1173	1424.0	9621			
17										1182	1454.0	**9404** *			
18										1183	1310.0	9506			

Table A8. *Cont.*

19										1189	1279.0	9481			
20										1202	1303.0	9252			
21										1266	1249.5	9639			
22										1284	**1198.5 ***	9588			

		ORB6			ORB7			ORB8			ORB9			ORB10	
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1090 ***	1382.5	9489	**433 ***	226.0	3813	**1016 ***	1919.5	8465	**1009 ***	1646.5	9402	**1055 ***	1366.5	9211
2	1091	1284.5	9341	437	225.0	3770	1025	1635.5	**8181 ***	1013	1595.0	9331	1065	790.5	8899
3	1134	1078.0	9177	439	271.5	3707	1047	1617.0	8457	1016	1534.0	9251	1108	843.0	8834
4	1153	1059.0	9182	453	220.0	3742	1148	1575.0	8319	1027	1644.0	9187	1114	**686.5 ***	8810
5	1168	969.0	**9030 ***	465	236.0	3697	1150	1564.0	8312	1036	1669.0	9130	1115	687.5	8795
6	1204	945.0	9072	471	**173.5 ***	**3620 ***	1176	1565.0	8294	1043	1479.0	9206	1246	1080.0	**8747 ***
7	1221	**907.0 ***	9034				1184	**1502.0 ***	8301	1063	1360.0	8975			
8										1064	**1355.0 ***	8966			
9										1066	1378.0	8942			
10										1073	1358.5	8956			
11										1083	1426.0	**8885 ***			
12										1092	1417.0	8914			

Table A9. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [42].

		LA01			LA02			LA03			LA04			LA05	
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**666 ***	1416.0	5550	**663 ***	1327.5	5145	**617 ***	1807.5	5353	**598 ***	1396.0	5096	**593 ***	1241.5	4601
2	666	1367.0	5561	677	1284.0	5053	624	1516.0	4890	598	1414.0	5094	593	1240.5	4604
3	666	1444.0	5500	685	**925.0 ***	**4805 ***	630	1444.0	4982	602	1181.0	4842	593	1290.0	4516
4	666	1325.5	5577				630	1511.5	4977	610	1049.0	4730	596	1277.0	4583
5	667	1465.5	5488				633	1383.5	4816	644	1083.5	**4726 ***	597	1242.0	4537
6	668	1269.0	5403				637	1345.5	4820	660	**1014.0 ***	4743	600	1233.5	4546
7	672	1245.5	5468				650	**1147.5 ***	4673	660	1027.5	4737	600	1273.0	4499
8	674	1246.0	5396				673	1164.0	**4632 ***				600	1190.5	4553
9	676	1313.0	5348										603	1162.0	4571
10	702	1229.5	5438										607	1154.5	4518
11	706	1099.5	5177										607	1185.0	4497
12	726	1072.5	5210										608	1176.5	4502
13	764	**1001.0 ***	**5176 ***										610	1133.5	4502
14													613	**1093.0 ***	4502
15													614	1130.5	4494
16													622	1164.0	4459
17													648	1209.0	4424
18													650	1198.0	**4413 ***
		LA06			**LA07**			**LA08**			**LA09**			**LA10**	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**926 ***	4193.5	10151	**890 ***	4398.0	10014	**863 ***	3719.5	9421	**951 ***	4212.5	10607	**958 ***	4562.0	10536
2	927	4150.5	10108	893	4494.0	9908	870	3644.5	9346	954	4387.0	10601	958	4558.5	10587
3	943	4104.0	10028	894	4092.5	9651	896	**3401.5 ***	**9139 ***	960	4284.5	10586	960	4507.0	10481
4	964	4061.5	9978	904	**3890.5 ***	**9452 ***				966	**4077.0 ***	**10411 ***	965	4277.0	10251
5	992	**4034.5 ***	**9951 ***										988	**4271.0 ***	**10,245 ***

Table A9. *Cont.*

	LA11			LA12			LA13			LA14			LA15		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1222** *	9579.5	17,606	**1039** *	7550.0	14,564	**1150** *	8618.0	16,397	**1292** *	9927.5	17,940	**1207** *	9792.5	17,960
2	1234	9317.5	17,344	1045	7514.0	14,528	1150	8641.5	16,377	1292	9966.0	17,847	1209	9679.5	17,847
3	1238	**9222.5** *	**17,249** *	1050	7498.0	14,512	1152	8608.0	16,387	1298	9919.5	17,857	1217	9644.5	17,812
4				1081	**7318.0** *	**14,332** *	1153	8459.5	16,160	1321	**9697.0** *	**17,716** *	1217	9692.5	17,769
5							1182	7884.0	15,577				1218	9628.5	17,705
6							1189	**7811.0** *	**15,504** *				1219	**9312.5** *	**17,336** *

	LA16			LA17			LA18			LA19			LA20		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**982** *	909.5	8738	**825** *	1045.0	7819	**872** *	609.5	7920	**901** *	569.0	8258	**938** *	749.0	8616
2	1008	771.0	8567	830	1016.0	7782	874	560.5	7836	904	398.0	8071	967	697.0	8549
3	1065	613.5	8503	848	1001.0	7698	905	555.0	8017	916	375.0	8146	967	695.0	8561
4	1082	603.0	8227	850	969.0	7569	908	555.5	7880	916	422.0	7972	969	674.0	8498
5	1091	**490.5** *	8311	854	983.0	7557	922	549.0	8056	921	342.0	7903	972	645.5	8578
6	1107	524.0	**8130** *	856	883.5	7656	930	549.0	7866	929	**276.0** *	7766	972	647.5	8470
7				865	845.5	7612	933	472.0	**7797** *	931	325.0	7765	978	558.0	8318
8				873	758.0	7517	933	**468.5** *	7824	953	488.0	**7759** *	1010	**531.0** *	8291
9				883	764.5	7500							1025	662.5	8277
10				894	752.0	7539							1041	612.0	**8069** *
11				911	758.0	7448									
12				918	723.0	7415									
13				927	775.0	**7336** *									
14				981	760.0	7384									
15				995	770.0	7373									
16				1009	730.0	7368									
17				1176	**720.0** *	7605									

	LA21			LA22			LA23			LA24			LA25		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1154** *	3406.5	15,329	**1041** *	3315.0	14,265	**1115** *	2616.5	14,458	**1047** *	2511.0	14,081	**1073** *	3252.0	14,388
2	1172	3329.5	15,084	1050	3118.0	14,068	1118	2599.5	14,441	1052	2477.0	14,047	1087	3217.0	14,315
3	1174	3035.5	14,835	1053	3035.0	14,000	1158	2459.0	14,476	1054	2870.5	14,001	1088	3143.0	14,241
4	1177	3059.5	14,607	1070	2994.0	13,975	1160	2457.0	14,436	1060	2613.5	13,860	1110	2638.0	13,761
5	1202	3044.5	14,763	1079	2754.0	13,625	1160	2722.5	14,389	1070	2593.5	13,918	1147	2633.0	13,793
6	1204	3024.5	14,743	1081	**2699.0** *	**13,562** *	1163	2437.0	14,416	1073	2598.5	13,874	1148	2682.5	**13,742** *
7	1220	3032.5	14,609				1172	2761.5	14,370	1079	2547.5	13,859	1148	**2623.5** *	13,764
8	1238	2881.5	14,783				1178	**2408.0** *	14,384	1080	2473.0	14,063			
9	1239	2877.5	14,666				1210	2595.5	14,373	1080	2546.5	**13,858** *			
10	1253	2832.5	14,696				1216	2562.5	**14,340** *	1087	**2368.0** *	13,911			
11	1347	2973.5	14,634												
12	1349	2883.0	14,507												
13	1356	2943.0	14,494												
14	1393	2936.0	14,419												
15	1393	2929.5	14,489												
16	1403	**2766.5** *	**14,412** *												

	LA26			LA27			LA28			LA29			LA30		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**1300** *	7356.5	23,129	**1374** *	8083.0	24,331	**1325** *	7440.0	23,463	**1328** *	8518.0	23,291	**1455** *	9085.0	25,105
2	1336	7171.5	22,944	1377	7946.0	24,194	1326	7315.0	23,338	1337	8513.0	23,286	1457	9071.0	25,091

Table A9. *Cont.*

3	1337	7077.5	22,850	1378	7660.0	23,875	1340	7233.0	23,256	1345	8501.0	23,274	1465	9211.5	25,064
4	1343	7047.5	22,820	1380	7641.0	23,856	1354	7185.0	23,176	1353	8534.0	23,273	1477	9196.5	25,049
5	1344	6971.5	22,744	1394	7645.5	23,854	1357	7096.0	23,087	1358	8464.0	23,203	1479	8374.5	24,204
6	1353	**6947.5 ***	22,720	1398	7494.0	23742	1360	7056.0	23,047	1360	8091.5	22,985	1481	8348.5	24,178
7	1396	7083.0	22,666	1401	7438.0	23,686	1375	6997.0	22,885	1363	8064.5	22,958	1519	8280.5	242,20
8	1454	7072.5	**22,660 ***	1402	7374.0	23,622	1384	6906.0	22,794	1368	8062.5	22,956	1543	8227.5	24167
9				1405	7408.5	23,586	1396	6674.5	22,672	1389	8208.0	22,939	1584	8391.5	24,097
10				1412	7327.0	23,575	1412	6568.5	22,566	1403	7990.5	22,836	1598	8090.5	23,796
11				1446	7265.0	23,513	1417	6518.5	22,509	1432	7971.5	22,865	1657	**7980.5 ***	**23,686 ***
12				1454	7367.0	23,500	1436	**6491.5 ***	**22,482 ***	1448	7972.0	22,776			
13				1469	7264.5	23,511				1453	7805.0	22,609			
14				1476	7228.0	23,476				1475	7733.5	22,627			
15				1483	7185.0	23,433				1525	**7664.5 ***	**22,558 ***			
16				1502	7226.5	23,352									
17				1602	**7109.5 ***	**23,312 ***									
		LA31			**LA32**			**LA33**			**LA34**			**LA35**	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1784 ***	219,44.5	44,731	**1850 ***	22,413.0	47,111	**1719 ***	22,284.5	44,738	**1768 ***	23,263.5	46,275	**1899 ***	24,702.5	47,930
2	1800	21,424.5	44,211	1850	22,411.5	47,265	1720	21,944.5	44,398	1774	22,903.5	45,915	1908	24,515.5	47,743
3	1807	21,363.5	44,150	1857	22,085.5	46,939	1722	21,802.5	44,256	1775	22,881.5	45,893	1909	23,489.5	46,717
4	1842	20,988.5	43,775	1859	22,074.5	46,928	1723	21,777.5	44,190	1776	22,657.5	45,669	1917	23,481.5	46,709
5	1843	**20,814.5 ***	**43,601 ***	1881	21,988.5	46,842	1734	21,723.5	44,177	1792	22,656.5	45,668	1919	23,379.5	46,607
6				1884	21,985.5	46,839	1743	21,447.5	43,901	1796	22,150.5	45,162	1923	**23,368.5 ***	46,596
7				1896	21,958.5	46,812	1746	21,446.5	43,900	1803	22,109.5	45,121	2029	23,393.5	**46,568 ***
8				1897	21,509.5	46,363	1750	21,134.5	43,508	1813	21,889.5	44,901			
9				1916	21,481.5	46,335	1755	21,040.5	43,414	1817	21,797.5	44,809			
10				2051	21,401.5	46,255	1771	21,024.5	43,478	1820	21,749.5	44,761			
11				2068	21,362.5	46,216	1776	20,995.5	43,449	1823	**21,740.5 ***	**44,752 ***			
12				2084	**21,294.5 ***	46,148	1777	20,945.5	43,399						
13				2148	21,372.5	**46,059 ***	1783	20,842.5	43,296						
14							1785	20,778.5	43,232						
15							1787	20,722.5	43,176						
16							1789	20,358.0	42,706						
17							1796	20,310.0	42,658						
18							1800	20,044.0	42,360						
19							1801	19,567.0	41,883						
20							1805	**19,558.0 ***	**41,874 ***						
		LA36			**LA37**			**LA38**			**LA39**			**LA40**	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1467 ***	3203.0	20,649	**1652 ***	2988.5	21,540	**1446 ***	2646.0	19,043	**1474 ***	2876.0	20,077	**1438 ***	2444.0	19,398
2	1503	3180.0	20,626	1653	2988.5	21,536	1472	2601.0	19,159	1494	2872.0	20,073	1531	2369.0	19,333
3	1515	3076.0	20,420	1656	2912.5	21,460	1473	2060.5	18,322	1513	2385.5	19,216	1561	**2336.0 ***	**19,300 ***
4	1519	3024.0	20,254	1691	3256.0	21,323	1491	**2000.5 ***	**18,262 ***	1597	2396.0	19,175			
5	1596	2988.5	20,597	1692	2894.0	21,493				1603	2362.0	19,101			
6	1616	2948.5	20,557	1696	3233.0	21,300				1605	**2254.0 ***	**18,993 ***			
7	1622	2868.5	20,477	1705	2757.0	21,254									
8	1632	2884.5	20,163	1751	2798.5	21,208									
9	1678	2903.5	20,106	1756	2888.5	21,064									
10	1704	2958.0	20,037	1757	2850.0	**21,005 ***									
11	1709	2869.0	19,948	1839	2670.5	21,086									
12	1735	2654.0	19,510	1883	**2578.5 ***	21,291									
13	1738	**2650.0 ***	**19,506 ***												

Table A10. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [43].

		ABZ5			ABZ6			ABZ7			ABZ8			ABZ9	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1296** *	565.0	11,621	**991** *	587.5	8826	**796** *	3124.0	14,127	**821** *	3504.0	14,883	**837** *	3263.0	14,378
2	1306	692.5	11,581	999	460.5	8658	797	2923.5	13,906	823	3447.0	14,826	845	2996.5	14,126
3	1321	683.5	11,572	1013	300.0	8753	803	2805.5	13,826	824	3428.0	14,807	848	2967.5	14,097
4	1322	523.0	11,801	1021	469.5	**8543** *	876	2684.5	13,608	825	3423.0	14,802	853	2936.5	14,066
5	1333	507.0	12,016	1037	407.5	8719	890	**2636.5** *	**13,556** *	835	**2786.0** *	14,111	856	**2900.5** *	**14,030** *
6	1334	407.5	11,786	1037	439.0	8674				847	2817.0	**14,086** *			
7	1334	403.0	11,861	1045	235.5	8614									
8	1337	574.0	11,604	1089	**197.5** *	8812									
9	1338	566.0	11,534	1115	203.5	8768									
10	1351	533.5	11,768												
11	1356	557.5	11,750												
12	1383	745.0	11,520												
13	1385	759.5	11,401												
14	1386	679.5	11,336												
15	1387	475.0	11,545												
16	1397	468.0	11,538												
17	1409	**407.0** *	**11,374** *												

Table A11. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [44].

		YN01			YN02			YN03			YN04	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1160** *	3154.5	20,470	**1155** *	3592.0	21,112	**1138** *	2732.5	19,941	**1225** *	4078.0	22,098
2	1166	2654.0	19,808	1159	3545.0	21,105	1154	2543.0	19,839	1228	3780.0	21,449
3	1188	2618.0	19,929	1165	3569.0	21,089	1158	2457.0	19,753	1231	3475.0	21,490
4	1193	2617.0	19,771	1166	3537.0	21,057	1204	2394.5	19,438	1232	3460.0	21,465
5	1197	2399.5	19,912	1169	3491.0	21,011	1223	2370.5	**19,414** *	1233	3745.0	21,414
6	1200	2220.5	19,745	1188	3171.5	20,606	1277	**2194.0** *	19,462	1245	3530.0	21,431
7	1201	**2114.0** *	**19,570** *	1211	3068.0	20,216				1247	3254.5	21,188
8				1212	3055.0	**20,203** *				1273	3236.5	21,170
9				1280	**3024.0** *	20,592				1286	3233.5	21,167
10										1325	**3169.0** *	**20,977** *

Table A12. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [30].

		TA01			TA11			TA21			TA31	
	MKS	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**	**MKS**	**TDS**	**FLT**
1	**1469** *	2284.0	19,027	**1649** *	7293.0	28,872	**2098** *	8414.5	38,534	**2126** *	21,558.0	55,423
2	1502	2201.0	19,461	1655	7264.0	28,843	2103	7979.0	38,146	2127	21,553.0	55,453
3	1515	1792.5	18,791	1672	7049.0	28,696	2113	7971.0	38,138	2135	21,552.0	55,417
4	1519	1783.5	18,801	1673	7045.0	28,692	2125	7247.5	37,366	2156	21,540.0	55405
5	1530	**1713.0** *	18,750	1677	6903.5	28,431	2128	7153.0	37,398	2161	21,416.0	55,316
6	1532	1725.0	**18,714** *	1696	6383.5	28,054	2137	6999.0	37,244	2173	21,109.0	55,009
7				1809	**6347.5** *	**28,018** *	2139	6974.0	37,209	2177	21052.0	54,952
8							2148	6820.5	37,028	2187	19,966.0	53,866
9							2150	6802.5	37,021	2205	**19,963.0** *	**53,863** *
10							2214	6550.0	36,679			
11							2238	6539.0	36,668			
12							2372	6316.0	36,317			
13							2373	**6190.0** *	**36,191** *			

Table A12. *Cont.*

	TA41			TA51			TA61			TA71		
	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT	MKS	TDS	FLT
1	**2632** *	21,027.5	67,904	**3128** *	73,001.0	129,878	**3420** *	74,932.0	151,378	**6094** *	366,221.5	517,558
2	2650	20,910.5	67,829	3132	72,689.0	129,566	3421	73956.0	150,402	6095	365,726.5	517,063
3	2666	20,826.5	67,745	3137	72,651.0	129,528	3423	73884.0	150,330	6098	365,546.5	516,883
4	2672	20,766.5	67,685	3192	70,022.5	126,809	3461	69,778.0	146,224	6174	**365,320.5** *	**516,657** *
5	2771	20,304.5	67,222	3249	**69,935.5** *	**126,722** *	3462	69,767.0	146,213			
6	2776	**20,265.5** *	**67,183** *				3478	**69,754.0** *	**146,200** *			

References

1. Coello, C.; Cruz, N. Solving Multiobjective Optimization Problems Using an Artificial Immune System. *Genet. Program. Evolvable Mach.* **2005**, *6*, 163–190. [CrossRef]
2. Garey, M.R.; Johnson, D.S.; Sethi, R. PageRank: The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* **1976**, *1*, 117–129. [CrossRef]
3. Ojstersek, R.; Brezocnik, M.; Buchmeister, B. Multi-objective optimization of production scheduling with evolutionary computation: A review. *Int. J. Ind. Eng. Comput.* **2020**, *11*, 359–376. [CrossRef]
4. Pinedo, M. *Scheduling: Theory, Algorithms, and Systems*; Springer: New York, NY, USA, 2008.
5. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Am. Assoc. Adv. Sci.* **1983**, *220*, 671–680. [CrossRef] [PubMed]
6. Dueck, G.; Scheuer, T. Threshold Accepting: A General Purpose Algorithm Appearing Superior to Simulated Annealing. *J. Comput. Phys.* **1990**, *90*, 161–175. [CrossRef]
7. Scaria, A.; George, K.; Sebastian, J. An artificial bee colony approach for multi-objective job shop scheduling. *Procedia Technol.* **2016**, *25*, 1030–1037. [CrossRef]
8. Méndez-Hernández, B.; Rodriguez Bazan, E.D.; Martinez, Y.; Libin, P.; Nowe, A. A Multi-Objective Reinforcement Learning Algorithm for JSSP. In Proceedings of the 28th International Conference on Artificial Neural Networks, Munich, Germany, 17–19 September 2019; pp. 567–584. [CrossRef]
9. López, A.; Coello, C. Study of Preference Relations in Many-Objective Optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO' 2009), Montreal, QC, Canada, 8–12 July 2009; pp. 611–618. [CrossRef]
10. Blasco, X.; Herrero, J.; Sanchis, J.; Martínez, M. *Decision Making Graphical Tool for Multiobjective Optimization Problems*; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4527, pp. 568–577. [CrossRef]
11. García-León, A.; Dauzère-Pérès, S.; Mati, Y. An Efficient Pareto Approach for Solving the Multi-Objective Flexible Job-Shop Scheduling Problem with Regular Criteria. *Comput. Oper. Res.* **2019**, *108*. [CrossRef]
12. Qiu, X.; Lau, H.Y.K. An AIS-based hybrid algorithm for static job shop scheduling problem. *J. Intell. Manuf.* **2014**, *25*, 489–503. [CrossRef]
13. Kachitvichyanukul, V.; Sitthitham, S. A two-stage genetic algorithm for multi-objective job shop scheduling problems. *J. Intell. Manuf.* **2011**, *22*, 355–365. [CrossRef]
14. Zhao, F.; Chen, Z.; Wang, J.; Zhang, C. An improved MOEA/D for multi-objective job shop scheduling problem. *Int. J. Comput. Integr. Manuf.* **2016**, *30*, 616–640. [CrossRef]
15. González, M.; Oddi, A.; Rasconi, R. Multi-objective optimization in a job shop with energy costs through hybrid evolutionary techniques. In Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, Pittsburgh, PA, USA, 18–23 June 2017; pp. 140–148.
16. Serafini, P. Simulated Annealing for Multi Objective Optimization Problems. In Proceedings of the Tenth International Conference on Multiple Criteria Decision Making, Taipei, Taiwan, 19–24 July 1992.
17. Bandyopadhyay, S.; Saha, S.; Maulik, U.; Deb, K. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. *Evol. Comput. IEEE Trans.* **2008**, *12*, 269–283. [CrossRef]
18. Liu, Y.; Dong, H.; Lohse, N.; Petrovic, S.; Gindy, N. An Investigation into Minimising Total Energy Consumption and Total Weighted Tardiness in Job Shops. *J. Clean. Prod.* **2013**, *65*, 87–96. [CrossRef]
19. Zitzler, E.; Thiele, L. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Trans. Evol. Comput.* **2000**, *3*, 257–271. [CrossRef]
20. Wisittipanich, W.; Kachitvichyanukul, V. An Efficient PSO Algorithm for Finding Pareto-Frontier in Multi-Objective Job Shop Scheduling Problems. *Ind. Eng. Manag. Syst.* **2013**, *12*, 151–160. [CrossRef]
21. Lei, D.; Wu, Z. Crowding-measure-based multiobjective evolutionary algorithm for job shop scheduling. *Int. J. Adv. Manuf. Technol.* **2006**, *30*, 112–117. [CrossRef]
22. Kurdi, M. An Improved Island Model Memetic Algorithm with a New Cooperation Phase for Multi-Objective Job Shop Scheduling Problem. *Comput. Ind. Eng.* **2017**, *111*, 183–201. [CrossRef]

23. Méndez-Hernández, B.; Ortega-Sánchez, L.; Rodriguez Bazan, E.D.; Martinez, Y.; Fonseca-Reyna, Y. Bi-objective Approach Based in Reinforcement Learning to Job Shop Scheduling. *Revista Cubana de Ciencias Informáticas* **2017**, *11*, 175–188.
24. Aarts, E.H.L.; van Laarhoven, P.J.M.; Lenstra, J.K.; Ulder, N.L.J. A Computational Study of Local Search Algorithms for Job Shop Scheduling. *INFORMS J. Comput.* **1994**, *6*, 118–125. [CrossRef]
25. Ponnambalam, S.G.; Ramkumar, V.; Jawahar, N. A multiobjective genetic algorithm for job shop scheduling. *Prod. Plan. Control* **2001**, *12*, 764–774. [CrossRef]
26. Suresh, R.K.; Mohanasundaram, M. Pareto archived simulated annealing for job shop scheduling with multiple objectives. *Int. J. Adv. Manuf. Technol.* **2006**, *29*, 184–196. [CrossRef]
27. Zitzler, E.; Deb, K.; Thiele, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef] [PubMed]
28. Karimi, N.; Zandieh, M.; Karamooz, H. Bi-objective group scheduling in hybrid flexible flowshop: A multi-phase approach. *Expert Syst. Appl.* **2010**, *37*, 4024–4032. [CrossRef]
29. Deb, K.; Agrawal, S.; Pratap, A.; Meyarivan, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving from Nature*; Spring: Berlin/Heidelberg, Germany, 2000; Volume 1917.
30. Taillard, E. Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **1993**, *64*, 278–285. [CrossRef]
31. Deb, K. *Multiobjective Optimization Using Evolutionary Algorithms*; Wiley: New York, NY, USA, 2001.
32. Schott., J.R. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.
33. Veldhuizen, D.A.V. Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ph.D. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Dayton, OH, USA, 1999.
34. Sawaragi, Y.; Nakagama, H.; Tanino, T. *Theory of Multi-Objective Optimization*; Springer: Boston, MA, USA, 1985.
35. Bakuli, D.L. A Survey of Multi-Objective Scheduling Techniques Applied to the Job Shop Problem (JSP). In *Applications of Management Science: In Productivity, Finance, and Operations*; Emerald Group Publishing Limited: Bingley, UK, 2015; pp. 51–62.
36. Baker, K.R. Sequencing rules and due-date assignments in job shop. *Manag. Sci.* **1984**, *30*, 1093–1104. [CrossRef]
37. Sanvicente, S.H.; Frausto, J. A method to establish the cooling scheme in simulated annealing like algorithms. In Proceedings of the International Conference on Computational Science and Its Applications, Assisi, Italy, 14–17 May 2004; pp. 755–763.
38. Solís, J.F.; Sánchez, H.S.; Valenzuela, F.I. ANDYMARK: An analytical method to establish dynamically the length of the Markov chain in simulated annealing for the satisfiability problem. *Lect. Notes Comput. Sci.* **2006**, *4247*, 269–276.
39. May, R. Simple Mathematical Models With Very Complicated Dynamics. *Nature* **1976**, *26*, 457. [CrossRef] [PubMed]
40. Fisher, H.; Thompson, G.L. Probabilistic learning combinations of local job-shop scheduling rules. *Ind. Sched.* **1963**, *1*, 225–251.
41. Applegate, D.; Cook, W. A computational study of the job-shop scheduling problem. *ORSA J. Comput.* **1991**, *3*, 149–156. [CrossRef]
42. Lawrence, S. *Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques (Supplement)*; Graduate School of Industrial Administration, Carnegie-Mellon University: Pittsburgh, PA, USA, 1984.
43. Adams, J.; Balas, E.; Zawack, D. The shifting bottleneck procedure for job shop scheduling. *Manag. Sci.* **1988**, *34*, 391–401. [CrossRef]
44. Yamada, T.; Nakano, R. A genetic algorithm applicable to large-scale job-shop problems. In Proceedings of the Second International Conference on Parallel Problem Solving from Nature, Brussels, Belgium, 28–30 September1992; pp. 281–290.
45. Hansen, P.B. Simulated Annealing. In *Electrical Engineering and Computer Science-Technical Reports*; School of Computer and Information Science, Syracuse University: Syracuse, NY, USA, 1992.
46. Tripathi, P.K.; Bandyopadhyay, S.; Pal, S.K. Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients. *Inf. Sci.* **2007**, *177*, 5033–5049. [CrossRef]