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 Cuauhnahuac 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

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 experimented with 58 benchmark instances, and they used 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.

| 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].

\[
MID = \frac{\sum_{i=1}^{Q} c_i}{Q}
\]

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].

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

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 \( \bar{d} = \frac{\sum_{i=1}^{Q} d_i}{Q} \) and \( d_i = \min \{|f_{1,i}(x) - f_{1,j}(x)| + |f_{2,i}(x) - f_{2,j}(x)| + \cdots + |f_{M,i}(x) - f_{M,j}(x)|\} \), 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.

\[
HV = \text{volume}\left(\bigcup_{i=1}^{Q} v_i\right)
\]  

(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].

\[
\Delta = \frac{\sum_{k=1}^{M} \hat{d}_k^T + \sum_{i=1}^{Q} |d_i - \hat{d}|}{\sum_{k=1}^{M} \hat{d}_k^T + Q \times \bar{d}}
\]  

(4)

where \(\hat{d}_k^T\) 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, \(\hat{d}\) 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].

\[
IGD(Q) = \left(\frac{\sum_{j=1}^{T} d_j^p}{|T|}\right)^{1/p}
\]  

(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).

\[
d_j = \min_{q=1}^{Q} \sqrt{\sum_{m=1}^{M} (fm(t_j) - fm(q))^2}
\]  

(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).

\[
C(A, B) = \frac{\{b \in B | \exists a \in A : a \leq b\}}{|B|}
\]  

(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]:

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

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:

\[
t_j \geq t_i + p_i \quad \text{For all } i, j \in O \text{ when } i \text{ precedes } j
\]

\[
t_j \geq t_i + p_i \text{ or } t_i \geq t_j + p_j \quad \text{For all } i, j \in O \text{ when } M_i = M_j
\]

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 \( O_{ij} \) (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 \( D_j = \tau \sum_{i=1}^m p_{ij} \) [36], where \( p_{ij} \) 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 (\( \tau \)), 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 [3]. 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 \((T_{\text{initial}})\) and final \((T_{\text{final}})\) temperatures, the alpha value \((\alpha)\) for decreasing the temperature, and beta \((\beta)\) for increasing the length of the Metropolis cycle. The current temperature \((T_k)\) is set in line 2. An initial solution \((s_{\text{current}})\) 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 \((T_{\text{final}})\). The Metropolis cycle starts in line 5, where a neighboring solution \((s_{\text{new}})\) is generated (line 6). In line 7 the increment \(\Delta E\) of the objective function is determined for the current solution \((s_{\text{current}})\) and the new one \((s_{\text{new}})\). 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\((T_{\text{initial}}, T_{\text{final}}, \alpha, \beta, L_k)\)
2: \(T_k \leftarrow T_{\text{initial}}\)
3: \(s_{\text{current}} \leftarrow \text{RandomInitialSolution()}\)
4: while \(T_k \geq T_{\text{final}}\) do
5:     for 1 to \(L_k\) do
6:         \(s_{\text{new}} \leftarrow \text{perturbation}(s_{\text{current}})\)
7:         \(\Delta E \leftarrow E(s_{\text{new}}) - E(s_{\text{current}})\)
8:         if \(\Delta E < 0\) then
9:             \(s_{\text{current}} \leftarrow s_{\text{new}}\)
10:         else
11:             if \(e^{-\Delta E/T_k} > \text{random}(0, 1)\) then
12:                 \(s_{\text{current}} \leftarrow s_{\text{new}}\)
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 \(s_{\text{current}}\)
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 \(E_{\text{max}}\) and \(E_{\text{min}}\), respectively. These energies appeared in the Boltzmann distribution criterion that states that a bad solution is accepted in a temperature \(T\) when \(\text{random}(0, 1) \leq e^{-\Delta E/T}\). For JSSP, \(\Delta E\) 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 $\Delta E_{\text{max}}$ and $\Delta E_{\text{min}}$. 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 ($T_{\text{initial}}$): 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 $\Delta E_{\text{max}}$ and $\Delta E_{\text{min}}$. Then $P(\Delta E_{\text{max}})$, is the probability of accepting a solution with the maximum deterioration and it is calculated with (12). Thus the value of the initial temperature ($T_{\text{initial}}$) is calculated with (13).

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

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

Final temperature ($T_{\text{final}}$): 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 $P(\Delta E_{\text{min}})$, which is the probability of accepting a solution with minimum deterioration.

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

Alpha value ($\alpha$): 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).

$$T_{k+1} = \alpha T_k \quad \text{(15)}$$

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 $L_{\text{min}}$ is the value of $L_k$ at the initial temperature, and $L_{\text{max}}$ is the $L_k$ at the final temperature, then the Formula (16) is used.

$$L_{k+1} = \beta L_k \quad \text{(16)}$$

where $\beta$ 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, $T_{\text{final}}$ and $L_{\text{max}}$ are calculated with (17) and (18).

$$T_{\text{final}} = \alpha^n T_{\text{initial}} \quad \text{(17)}$$

$$L_{\text{max}} = \beta^n L_{\text{min}} \quad \text{(18)}$$

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

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

$$\beta = e^{\frac{\ln(L_{\text{max}}) - \ln(L_{\text{min}})}{n}} \quad \text{(20)}$$
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).

\[
P(S_j) = 1 - e^{-\frac{N}{|V_{si}|}} \quad (21)
\]

\[
N = -|V_{si}| \ln(1 - P(S_j)) = C |V_{si}| \quad (22)
\]

\[
C = \ln(P(S_j)) \quad (23)
\]

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

\[
L_{max} = N = C |V_{si}| \quad (24)
\]

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

5.3. Chaotic Multi-Objective Simulated Annealing (CMOSA)

As we previously mentioned, the AMOSA algorithm was proposed in \cite{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 \( (\alpha) \), beta \( (\beta) \), 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 \( \times \text{timesLS} = 10 \)).

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 $\beta$ parameter value. Additionally, in line 38, the current temperature ($T_k$) is decreased by multiplying it by the $\alpha$ 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 ← 10, counterTrapped ← 0, isCaught ← FALSE
3: iterationsLocalSearch ← tasks $\times$ timesLS, verifyCaught ← TRUE, countCaught ← 0
4: $mks_{current}, tds_{current}, f lt_{current} ← \text{calculateValues}(s_{current})$  \hfill $\triangleright$ mks : makespan, tds : tardiness, flt : flowtime
5: $T_k ← T_{initial}$
6: while $T_k \geq T_{final}$ do
7:     for $i ← 0$ to $L_k$ do
8:         if isCaught = TRUE then
9:             isCaught ← FALSE
10:        end if
11:        for $k ← 0$ to iterationsLocalSearch do
12:            if $k = 0$ then
13:                $s_{new} ← \text{chaoticPerturbation}(s_{current})$ \hfill $\triangleright$ See Algorithm 4
14:            else
15:                $s_{new} ← \text{regularPerturbation}(s_{current})$ \hfill $\triangleright$ Exchange of two operations
16:            end if
17:            $mks_{new}, tds_{new}, f lt_{new} ← \text{calculateValues}(s_{new})$
18:            if ($mks_{new} < mks_{current}$) AND ($tds_{new} < tds_{current}$) AND ($flt_{new} < flt_{current}$) then
19:                $s_{current} ← s_{new}$
20:            end if
21:        end for
22:        else
23:            $s_{new} ← \text{regularPerturbation}(s_{current})$
24:            $mks_{new}, tds_{new}, f lt_{new} ← \text{calculateValues}(s_{new})$
25:        end if
26:        if ($mks_{new} \neq mks_{current}$) AND ($tds_{new} \neq tds_{current}$) AND ($flt_{new} \neq flt_{current}$) then
27:            verifyDominanceCMOSA($T_k, s_{new}, s_{current}$) \hfill $\triangleright$ See Algorithm 5
28:        end if
29:    end for
30:    if verifyCaught = TRUE then
31:        if caught($s_{current}, counterTrapped$) = TRUE then
32:            countCaught = countCaught + 1
33:        end if
34:        if countCaught = MAXSTAGNANT then
35:            verifyCaught ← FALSE
36:        end if
37:    end if
38:    $L_k ← \beta \times L_k$
39:    $T_k ← \alpha \times T_k$
40: end while
41: return $s_{current}$
42: 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\textsubscript{current}, counterTrapped)
2: isCaught ← FALSE, timesDominated ← 0, maxTrapped ← 10
3: timesDominated ← countTimesDominated(s\textsubscript{current})
4: if timesDominated = 0 then
5: \( F \leftarrow s\textsubscript{current} \)
6: end if
7: if timesDominated \( \geq 1 \) then
8: counterTrapped ← counterTrapped + 1
9: end if
10: if counterTrapped = maxTrapped then
11: isCaught ← TRUE
12: counterTrapped ← 0
13: end if
14: return isCaught
15: end procedure

In this Algorithm 3 the current solution \( s\textsubscript{current} \) and the counter of times it has trapped \( \text{counterTrapped} \) 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 \( \text{counterTrapped} \) is incremented (line 8). When \( \text{counterTrapped} \) equals the maximum number of trapped allowed (line 10), the value of \( \text{isCaught} \) 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:

\[
x_{n+1} = rx_n(1 - x_n)
\]  

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

```plaintext
1: procedure CHAOTICPERTURBATION(s_current)
2:     feasibleTasksNumber ← 0, r ← 4, repeat ← TRUE, X_n ← 0, X_{n1} ← 0
3:     while counter < tasks do
4:         feasibleTasksNumber ← searchFeasibleTasks()
5:         if feasibleTasksNumber = 1 then
6:             index ← 0
7:         else
8:             while repeat = TRUE do
9:                 X_n ← random(0, 1)
10:                for i ← 0 to feasibleTasksNumber do
11:                   X_{n1} ← (r × X_n) × (1.0 − X_n)
12:                   if X_{n1} > 0.5 then
13:                      index ← i
14:                      repeat ← FALSE
15:                      break
16:             end if
17:             X_n ← X_{n1}
18:         end while
19:     end if
20:     s_new ← addTask(index)
21:     counter ← counter + 1
22: 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

1: procedure VERIFYDOMINANCECMOSA(T_k, s_new, s_current, mks_new, tds_new, flt_new, mks_current, tds_current, flt_current)
2: newDominanceCurrent ← FALSE, currentDominanceNew ← FALSE
3: if s_new ≺ s_current then
4: s_current ← s_new,
5: newDominanceCurrent ← TRUE
6: end if
7: if s_current ≺ s_new then
8: ΔMKS ← mks_new - mks_current
9: ΔTDS ← tds_new - tds_current
10: ΔFLT ← flt_new - flt_current
11: Δ ← ΔMKS + ΔTDS + ΔFLT
12: if random(0, 1) < e^{-Δ/T_k} then
13: F ← s_current
14: s_current ← s_new
15: end if
16: currentDominanceNew ← TRUE
17: end if
18: if (newDominanceCurrent = FALSE) AND (currentDominanceNew = FALSE) then
19: F ← s_current
20: s_current ← s_new
21: end if
22: return s_current
23: 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 γ_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 $\gamma$ 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)

1: procedure CMOTA($T_{initial}, T_{final}, \alpha, \beta, L_k, s_{current}$)
2: counter ← 1, MAXSTAGNANT ← 10, counterTrapped ← 0, isCaught ← FALSE
3: iterationsLocalSearch ← tasks $\times$ timesLS, verifyCaught ← TRUE, countCaught ← 0
4: mks$_{current}$, tds$_{current}$, flt$_{current}$ ← calculateValues($s_{current}$) ✷ mks : makespan, tds : tardiness, flt : flowtime
5: $T_k$ ← $T_{initial}$
6: while $T_k \geq T_{final}$ do
7: for $i$ ← 0 to $L_k$ do
8: if isCaught = TRUE then
9: isCaught = FALSE
10: for $k$ ← 0 to iterationsLocalSearch do
11: if $k = 0$ then
12: $s_{new}$ ← chaoticPerturbation($s_{current}$) ✷ See Algorithm 4
13: else
14: $s_{new}$ ← regularPerturbation($s_{current}$) ✷ Exchange of two operations
15: end if
16: mks$_{new}$, tds$_{new}$, flt$_{new}$ ← calculateValues($s_{new}$)
17: if (mks$_{new}$ < mks$_{current}$) AND (tds$_{new}$ < tds$_{current}$) AND (flt$_{new}$ < flt$_{current}$) then
18: $s_{current}$ ← $s_{new}$
19: end if
20: end for
21: else
22: $s_{new}$ ← regularPerturbation($s_{current}$)
23: mks$_{new}$, tds$_{new}$, flt$_{new}$ ← calculateValues($s_{new}$)
24: end if
25: if (mks$_{new}$ $\not=$ mks$_{current}$) AND (tds$_{new}$ $\not=$ tds$_{current}$) AND (flt$_{new}$ $\not=$ flt$_{current}$) then
26: verifyDominanceCMOTA(counter, $T_k$, $s_{new}$, $s_{current}$) ✷ See Algorithm 7
27: end if
28: end for
29: if verifyCaught = TRUE then
30: if caught($s_{current}$, counterTrapped) = TRUE then
31: countCaught = countCaught + 1
32: if countCaught = MAXSTAGNANT then
33: verifyCaught ← FALSE
34: end if
35: end if
36: $L_k$ ← $\beta \times L_k$
37: $T_k$ ← $\alpha \times T_k$
38: counter ← counter + 1
39: end while
40: return $s_{current}$
41: end procedure
Algorithm 7 Verify dominance CMOTA

1: procedure VERIFYDOMINANCECMOTA(counter, T_k, s_new, s_current)
2: \( \gamma \leftarrow 1, \gamma_{reduced} \leftarrow 0.978, setT \leftarrow 1, \text{bound} \leftarrow \text{NumberOfTemperatures} \times \text{limit} \)
3: newDominanceCurrent \leftarrow \text{FALSE}, currentDominatesNew \leftarrow \text{FALSE}
4: if counter < bound then
5: \( T \leftarrow T_k \)
6: end if
7: if (counter = bound) AND (setT = 1) then
8: setT \leftarrow 0
9: \( T \leftarrow T_k \)
10: end if
11: if setT = 0 then
12: \( \gamma \leftarrow \gamma_{reduced} \)
13: end if
14: if s_new \( \prec \) s_current then
15: s_current \leftarrow s_new
16: newDominatesCurrent \leftarrow \text{TRUE}
17: end if
18: if s_current \( \prec \) s_new then
19: if random(0, 1) \( < \) T then
20: \( F \leftarrow s_{\text{current}} \)
21: s_current \leftarrow s_new
22: end if
23: currentDominatesNew \leftarrow \text{TRUE}
24: end if
25: if (newDominatesCurrent = \text{FALSE}) AND (currentDominatesNew = \text{FALSE}) then
26: \( F \leftarrow s_{\text{current}} \)
27: s_current \leftarrow s_new
28: end if
29: \( T \leftarrow T \times \gamma \)
30: 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 Ecatecl 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 |
|----------|-------------|-------------|------------------------|
| 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.
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 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.

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

| Instance | SPEA [8] | CMOEA [8] | MOPSO [8] | MOMARLA [8] | CMOSA | CMOTA |
|----------|---------|-----------|-----------|-------------|--------|-------|
| FT06     | 0.07    | 0.07      | 0.50      | 0.65        | 0.64   | 0.75  |
| FT10     | 0.17    | 0.26      | 0.87      | 0.96        | 0.71   | 0.69  |
| ABZ5     | 0.34    | 0.33      | 0.36      | 0.40        | 0.85   | 0.56  |
| ABZ6     | 0.22    | 0.36      | 0.31      | 0.42        | 0.60   | 0.81  |
| ABZ7     | 0.51    | 0.45      | 1.00      | 1.00        | 0.79   | 0.51  |
| LA26     | 0.33    | 0.39      | 0.47      | 0.47        | 0.91   | 0.70  |
| LA27     | 0.58    | 0.56      | 0.41      | 0.60        | 0.71   | 0.93  |
| ORB01    | 0.62    | 0.74      | 0.59      | 0.80        | 0.87   | 0.63  |
| ORB02    | 0.20    | 0.04      | 0.30      | 0.53        | 0.88   | 0.77  |
| ORB03    | 0.69    | 0.31      | 0.85      | 0.86        | 0.76   | 0.80  |
| ORB04    | 0.63    | 0.28      | 0.52      | 0.79        | 0.76   | 0.81  |
| 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, \( \Gamma \) 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 \( O(nT^2) \) and \( O(MT^2) \) 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 \( O(MT^2) \) [46]. Thus, the relation between them is \( np/T^2 \) 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   |
|-----------|---------|----------|----------|-----------|---------|---------|
| \( O(nT^2) \) | \( O(MT^2) \) | \( O(MT^2) \) | \( O(Mnp) \) | \( O(Mnp) \) | \( O(Mnp) \) |}

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, CMOE, 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 | 8890.0 | 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  |
| 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 | 911 | 744.5 | 9049 | 1030 | 746.5 | 7869 |
| 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 | 7260 * |      |      |      |      |      |
| 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  |
| 1               | 1097 | 1318 | 9573 | 423  | 207  |
| 2               | 1100 | 1199 | 9505 | 424  | 167  |
| 3               | 1100 | 1267 | 9434 | 431  | 161  |
| 4               | 1105 | 1225 | 9434 | 439  | 295  |
| 5               | 1105 | 1227 | 9412 | 427  | 204  |
| 6               | 1110 | 1255 | 9409 | 453  | 230  |
| 7               | 1113 | 1220 | 9452 | 455  | 204  |
| 8               | 1114 | 1078 | 9287 | 459  | 213  |
| 9               | 1114 | 1153 | 9109 | 461  | 216  |
| 10              | 1171 | 1097 | 9194 | 461  | 203  |
| 11              | 1191 | 1018 | 9145 | 461  | 186  |
| 12              | 1233 | 988  | 9225 | 466  | 202  |
| 13              | 466  | 171  | 361  |      |      |
| 14              | 470  | 184  | 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  |
| 1               | 666  | 1194 | 5436 | 655  | 1207 |
| 2               | 666  | 1237 | 5362 | 656  | 1161 |
| 3               | 667  | 1382 | 5357 | 665  | 1222 |
| 4               | 668  | 1068 | 5328 | 665  | 1203 |
| 5               | 668  | 1074 | 5309 | 671  | 1042 |
| 6               | 670  | 1269 | 5300 | 673  | 1094 |
| 7               | 672  | 1152 | 5260 | 681  | 938  |
| 8               | 688  | 1145 | 5247 | 695  | 927  |
| 9               | 700  | 1120 | 5297 | 695  | 930  |
| 10              | 706  | 1081 | 5241 | 696  | 910  |
| 11              | 706  | 1179 | 5225 | 714  | 997  |
| 12              | 713  | 1065 | 5203 | 715  | 936  |
| 13              | 718  | 1025 | 5235 | 736  | 925  |
| 14              | 727  | 1056 | 5138 | 741  | 993  |
| 15              | 734  | 1046 | 5184 | 771  | 909  |
| 16              | 743  | 1089 | 5101 |      |      |
| 17              | 751  | 951  | 5115 |      |      |
| 18              | 825  | 1098 | 5099 |      |      |
| 19              | 688  | 1087 | 4837 | 643  | 469  |
| 20              | 698  | 1055 | 4504 | 648  | 427  |
| 21              | 744  | 955  | 4382 | 650  | 455  |
| 22              | 744  | 891  | 4375 | 655  | 457  |
| 23              | 750  | 896  | 4323 | 663  | 455  |
| 24              | 757  | 867  | 4325 |      |      |
Table A3. Cont.

| LA06 | LA07 | LA08 | LA09 | LA10 |
|------|------|------|------|------|
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1    | 926  | 4185.5 | 10,142 | 890 * | 4066.5 | 9554 |
| 2    | 927  | 4183.0 | 10,171 | 890  | 4044.0 | 9496 |
| 3    | 929  | 4062.0 | 10,050 | 894  | 3974.5 | 9522 |
| 4    | 931  | 4122.0 | 10,041 | 896  | 3646.5 | 9387 |
| 5    | 938  | 3911.0 | 9870  | 904  | 3684.0 | 9248 |
| 6    | 940  | 3827.0 | 9786  | 906  | 3615.0 | 9219 |
| 7    | 949  | 3799.0 | 9199  | 974  | 3595.0 * | 9199 |
| 8    | 951  | 3944.5 | 951  | 976  | 4298.0 | 9266 |
| 9    | 958  | 4508.0 | 967  | 929  | 4313.0 | 9266 |

| LA11 | LA12 | LA13 | LA14 | LA15 |
|------|------|------|------|------|
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1    | 1222 | 9157.5 | 17,184 | 1039 * | 7218.0 | 14,229 |
| 2    | 1225 | 8947.5 | 16,853 | 1041 | 7203.0 | 14,167 |
| 3    | 1241 | 8879.5 | 16,785 | 1043 | 7198.0 | 14,229 |
| 4    | 1242 | 8862.5 | 16,768 | 1045 | 7164.0 | 14,162 |
| 5    | 1243 | 8860.5 | 16,766 | 1047 | 7126.0 | 14,229 |
| 6    | 1256 | 8811.5 | 16,798 | 1050 | 7114.0 | 14,229 |
| 7    | 1257 | 8725.5 | 16,712 | 1051 | 7126.0 | 14,162 |
| 8    | 1258 | 8765.5 | 16,671 | 1053 | 7070.0 | 14,167 |
| 9    | 1265 | 8650.5 * | 16,637 * | 1055 | 7070.0 | 14,162 |
| 10   | 1266 | 8650.5 * | 16,637 * | 1057 | 7070.0 | 14,167 |
| 11   | 1267 | 8725.5 | 16,712 | 1059 | 7070.0 | 14,167 |
| 12   | 1278 | 8765.5 | 16,671 | 1061 | 7070.0 | 14,167 |
| 13   | 1283 | 8638.5 | 16,802 | 1063 | 7070.0 | 14,167 |
| 14   | 1289 | 8638.5 | 16,802 | 1065 | 7070.0 | 14,167 |

| LA16 | LA17 | LA18 | LA19 | LA20 |
|------|------|------|------|------|
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1    | 968 * | 983.5 | 8777 | 796 * | 799.0 | 7502 |
| 2    | 982  | 904.0 | 8754 | 796  | 784.0 | 7509 |
| 3    | 988  | 898.5 | 8608 | 810  | 855.0 | 7492 |
| 4    | 992  | 882.0 | 8752 | 811  | 864.0 | 7492 |
| 5    | 994  | 816.5 | 8669 | 813  | 702.0 | 7458 |
| 6    | 1000 | 873.0 | 8570 | 813  | 745.0 | 7458 |
| 7    | 1003 | 900.0 | 8565 | 816  | 693.0 | 7478 |
| 8    | 1003 | 900.0 | 8545 | 820  | 630.0 | 7395 |
| 9    | 1003 | 942.0 | 8474 | 823  | 670.5 | 7334 |
| 10   | 1008 | 493.0 | 8205 | 824  | 633.5 | 7240 |
| 11   | 1016 | 553.5 | 8063 | 831  | 632.5 | 7281 |
| 12   | 1040 | 459.5 | 8232 | 833  | 625.5 | 7320 |
| 13   | 1050 | 352.0 | 7997 | 835  | 671.5 | 7203 |
| 14   | 1066 | 345.5 | 8285 | 836  | 596.5 | 7291 |
| 15   | 1071 | 341.5 | 8068 | 836  | 611.5 | 7284 |
| 16   | 1073 | 401.0 | 7980 | 840  | 597.0 | 7267 |
| 17   | 1095 | 326.5 * | 7908 * | 840 | 612.0 | 7260 |
| 18   | 842  | 612.0 | 7980 | 840  | 597.0 | 7267 |
| 19   | 849  | 522.0 | 7208 | 892  | 455.0 | 7519 * |
| 20   | 849  | 521.5 | 7232 | 892  | 439.0 | 7626 |
Table A3. Cont.

LA21 | LA22 | LA23 | LA24 | LA25
--- | --- | --- | --- | ---
MKS | TDS | FLT | 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 | 13,320 | 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 | 2738.5 | 13,552 | 1080 | 2173.5 | 14,152 | 1077 | 2010.5 | 13,458 | 1100 | 2756.5 | 13,951
4 | 1128 | 3137.5 | 14,868 | 1034 | 2565.5 | 13,375 | 1105 | 2231.5 | 14,149 | 1079 | 1981.5 | 13,390 | 1104 | 2764.5 | 13,940
5 | 1137 | 2998.5 | 14,400 | 1038 | 2462.5 | 13,253 | 1109 | 2243.5 | 14,147 | 1088 | 2145.5 | 13,320 | 1118 | 2721.0 | 13,962
6 | 1141 | 2892.5 | 14,636 | 1045 | 2392.5 | 13,169 | 1102 | 2137.5 | 14,103 | 1118 | 2768.0 | 13,938
7 | 1144 | 2821.5 | 14,565 | 1047 | 2696.5 | 13,510 | 1123 | 2618.5 | 13,658
8 | 1146 | 2939.0 | 14,346 | 1050 | 2614.5 | 13,445 | 1134 | 2536.5 | 13,577
9 | 1150 | 2543.0 | 14,344 | 1058 | 2565.5 | 13,396 | 1134 | 2529.0 | 13,770
10 | 1150 | 2639.5 | 14,316 | 1068 | 2544.5 | 13,375 | 1134 | 2517.5 | 13,553
11 | 1158 | 2545.5 | 14,222 | 1082 | 2462.5 | 13,253 | 1159 | 2457.0 | 13,654
12 | 1164 | 2511.5 | 14,188 | 1087 | 2392.5 | 13,169 | 1160 | 2451.5 | 13,666
13 | 1179 | 2393.5 | 14,204 | 1099 | 2332.5 | 13,109 | 1173 | 2530.0 | 13,470
14 | 1182 | 2331.5 | 14,165
15 | 1182 | 2355.5 | 14,153
16 | 1183 | 2454.5 | 14,131
17 | 1227 | 2328.0 | 14,238
18 | 1247 | 2225.0 | 14,161
19 | 1258 | 2561.5 | 13,967
20 | 1272 | 2527.5 | 13,963
21 | 1285 | 2465.5 | 13,871
22 | 1329 | 2454.5 | 13,131

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 | 25,294
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
7 | 1329 | 6560.5 | 22,333
8 | 1338 | 6616.5 | 22,129
9 | 1340 | 6510.5 | 22,276
10 | 1377 | 6307.0 | 21,940
11 | 1454 | 6429.0 | 22,298 | 1375 | 7398.5 | 22,289
12 | 1476 | 6239.0 | 22,013 | 1376 | 7464.5 | 22,218
13 | 1477 | 6141.0 | 21,915
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  |
| 1 | 1784 * | 20,830.5 | 43,617 | 1850 * | 20,861.5 | 45,715 | 1719 * | 20,933.5 | 43,387 | 1743 * | 22,605.5 | 45,617 |
| 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 |
| 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 |
| 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 |
| 5 | 1798 | 19,009.5 | 42,785 | 1889 | 20,059.5 | 44,913 | 1738 | 18,109.5 | 40,563 | 1759 | 21,041.5 | 44,037 |
| 6 | 1800 | 19,919.5 | *42,695* | 1900 | 20,049.5 | *44,903* | 1774 | 20,718.0 | 43,787 | 1901 | 22,481.5 | 45,709 |
| 7 | 1453 * | 3131.0 | 20,575 | 1569 * | 3065.0 | 21,444 | 1400 * | 1586.0 | 18,171 | 1444 * | 2371.0 | 19,447 |
| 8 | 1471 | 3030.5 | 20,309 | 1571 | 3077.0 | 21,436 | 1419 | 1578.5 * | 18,200 | 1452 | 2056.0 | 19,215 |
| 9 | 1474 | 2834.5 | 20,125 | 1574 | 3043.0 | 21,402 | 1421 | 2057.5 | 18,119 | 1498 | 1770.5 | 18,662 |
| 10 | 1475 | 2936.5 | 20,085 | 1574 | 3025.0 | 21,404 | 1439 | 2092.5 | 18,067 | 1499 | 1731.5 | 18,607 |
| 11 | 1476 | 2847.5 | 20,094 | 1580 | 3009.0 | 21,301 | 1468 | 1753.5 | 18,103 | 1504 | 1473.5 | 18,404 |
| 12 | 1476 | 2949.5 | 20,054 | 1584 | 3002.0 | 21,294 | 1473 | 1736.5 | 18,086 | 1621 | 1422.5 | 18,579 |
| 13 | 1487 | 2633.5 | 19,889 | 1590 | 2331.5 | 20,755 | 1496 | 1744.5 | 18,044 * | 1817 | 1902.0 * | 18,191 * |
| 14 | 1498 | 2474.5 | 19,694 | 1593 | 2289.5 | 20,748 | 1906 | 19,880.5 * | 42,892 * | 1907 | 2187.5 | 20,418 |
| 15 | 1505 | 2492.5 | 19,675 | 1608 | 2247.5 | 20,585 | 1700 | 2222.5 | 20,453 | 1920 | 2205.0 | 20,517 |
| 16 | 1529 | 2459.5 | 19,679 | 1618 | 2374.0 | 20,143 | 1700 | 2187.5 | 20,418 | 1781 | 2012.0 | 20,554 |
| 17 | 1530 | 2420.0 | 19,668 | 1621 | 2418.0 | 20,077 * | 1700 | 2012.0 | 20,554 | 1781 | 1964.5 | 20,634 |
| 18 | 1534 | 2335.5 | 19,812 | 1649 | 2234.5 | 20,600 | 1835.5 * | 20,309 | 19 | 1700 | 2187.5 | 20,418 |
| 19 | 1534 | 2472.5 | 19,650 | 1650 | 2237.5 | 20,587 | 1700 | 2222.5 | 20,453 | 1781 | 2012.0 | 20,554 |
| 20 | 1548 | 2278.5 | 19,755 | 1650 | 2241.5 | 20,557 | 1700 | 2222.5 | 20,453 | 1781 | 1964.5 | 20,634 |
| 21 | 1563 | 2015.5 * | 19,237 | 1700 | 2222.5 | 20,453 | 1700 | 2205.0 | 20,517 | 1781 | 1964.5 | 20,634 |
| 22 | 1573 | 2532.5 | 19,231 * | 1700 | 2205.0 | 20,517 | 1700 | 2205.0 | 20,517 | 1781 | 1964.5 | 20,634 |
### 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   |
| 1     | 1250  | *     | 145.0 | 11,006|       |       |       |       |       |       |       |
| 2     | 1250  | 134.0 | 11,025| 974   | 256.5 | 8524  | 753   | 2403.0| 13,257| 822   | 2846  |
| 3     | 1252  | 139.0 | 10,998| 974   | 251.5 | 8550  | 793   | 2305.0| 13,137| 805   | 3296  |
| 4     | 1289  | 141.0 | 10,984| 979   | 204.0 | 8464  |       | 773   | 2326.0| 822   | 2846  |
| 5     | 1289  | 142.0 | 10,946| 997   | 258.5 | 8357  |       | 775   | 2294.0| 833   | 2770  |
|       |       |       |       |       |       |       |       |       |       |       |       |

### 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             |
| 1               | 1103            | 2485.0          | 19,819          | 1133            | 2178.0          | 19,429          | 1083            | 2025.5          | 19,346          |
| 2               | 1105            | 2442.0          | 19,776          | 1137            | 2205.0          | 19,424          | 1084            | 2015.5          | 19,336          |
| 3               | 1105            | 2465.5          | 19,753          | 1140            | 2050.0          | 19,299          | 1084            | 2012.5          | 19,337          |
| 4               | 1106            | 2418.5          | 19,706          | 1140            | 2067.0          | 19,286          | 1089            | 2003.5          | 19,328          |
| 5               | 1106            | 2395.0          | 19,729          | 1148            | 2059.0          | 19,278          | 1090            | 1987.5          | 19,308          |
| 6               | 1108            | 1901.0          | 19,129          | 1150            | 1901.0          | 19,276          | 1138            | 2179.5          | 19,219          |
| 7               | 1111            | 1859.0          | 19,068          |                |                |                |                | 2157.5          | 18,751          |
| 8               | 1117            | 1867.0          | 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         | 1063           | 6409.5         | 27,903         | 2048           | 1083           | 7261.5         | 37,039         | 2083           | 50,557         |
| 2               | 1412           | 16,41.5        | 18,749         | 1607           | 6365.5         | 27,859         | 2050           | 6184.5         | 36,322         | 2091           | 50,404         |                |
| 3               | 1414           | 1809.0         | 18,704         | 1619           | 6051.5         | 27,722         | 2051           | 6184.5         | 36,290         | 2096           | 50,448         |                |
| 4               | 1433           | 1753.5         | 18,648         | 1750           | 6387.0         | 27,635         | 2074           | 6023.5         | 36,129         | 2097           | 50,112         |                |
| 5               | 1443           | 1733.5         | 18,739         | 1753           | 6307.0         | 27,555         | 2078           | 6017.5         | 36,123         | 2099           | 50,099         |                |
| 6               | 1448           | 1625.0         | 18,765         | 1766           | 6293.0         | 27,572         | 2091           | 6031.0         | 36,050         | 2106           | 19,879         |                |
| 7               | 1859           | 6088.0         | 27,679         | 2027           | 5939.0         | 35,462         |                |                |                | 2109           | 19,860         |                |
| 8               |                |                |                |                |                |                |                |                |                |                |                |                |
| 9               |                |                |                |                |                |                |                |                |                |                |                |                |
| 10              |                |                |                |                |                |                |                |                |                |                |                |                |
| 11              |                |                |                |                |                |                |                |                |                |                |                |                |
| 12              |                |                |                |                |                |                |                |                |                |                |                |                |
Table A6. Cont.

| TA41 | TA51 | TA61 | TA71 |
|------|------|------|------|
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1    | 2530 * | 18,610.5 | 65,529 | 3212 * | 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 | 70,885.0 | 147,131 | 6097 | 367,933.5 | 519,270 |
| 4    | 2733 | 18,257.0 | 65,116 | 3127 | 74,028.0 | 130,905 | 3571 | 71,034.0 | 147,318 | 6097 | 367,933.5 | 519,270 |
| 5    | 2736 | 18,228.0 | 65,087 | 3134 | 72,636.0 | 129,513 | 3589 | 70,685.0 | 147,608 | 6097 | 367,933.5 | 519,270 |
| 6    | 2743 | 18,197.0 | 65,056 | 3186 | 72,624.0 | 129,501 | 3597 | 70,550.0 * | 146,996 * | 6098 | 367,927.5 | 519,270 |
| 7    | 2832 | 18,128.5 | 65,047 | 3188 | 71,884.0 | 128,761 | 3661 | 70,685.0 | 147,608 | 6098 | 367,927.5 | 519,270 |
| 8    | 2949 | 17,853.5 * | 64,772 * | 3189 | 71,849.0 | 128,726 | 3677 | 70,685.0 | 147,608 | 6098 | 367,927.5 | 519,270 |
| 9    | 3202 | 70,643.0 | 127,520 | 6215 | 361,891.5 * | 519,270 |
| 10   | 3204 | 70,623.0 * | 127,500 * | 6215 | 361,891.5 * | 519,270 |

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

| FT06 | MKS  | TDS  | FLT  | FT10 | MKS  | TDS  | FLT  | FT20 | MKS  | TDS  | FLT  |
|------|------|------|------|------|------|------|------|------|------|------|------|
| ORB1 | ORB2 | ORB3 | ORB4 | ORB5 |
| 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 * | 9897 |
| 2    | 1190 | 1714.5 | 9619 | 983 | 971.5 | 8672 | 1134 | 1901.0 | 9579 | 1104 | 1720.5 | 10,062 |
| 3    | 1192 | 1721.5 | 985 | 985 | 913.5 | 8601 | 1208 | 1842.5 | 9770 | 1109 | 1695.5 | 10,117 |
| 4    | 1237 | 1787.5 | 9440 | 986 | 975.5 | 8593 | 1212 | 1795.5 | 9721 | 1111 | 1600.5 | 9865 |
| 5    | 1238 | 1714.5 | 9616 | 987 | 1009.0 | 8347 | 1217 | 1829.5 | 9698 | 1118 | 1507.0 | 9818 |
| 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 | 1138 | 1588.5 | 9768 |
| 10   | 1336 | 1415.5 | 9303 | 1015 | 854.5 | 8526 | 1143 | 1487.0 | 9798 | 1143 | 1487.0 | 9798 |
| 11   | 1339 | 1372.5 * | 9260 * | 1020 | 625.5 | 8251 | 1153 | 1626.0 | 9674 | 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. Non-dominated front obtained by CMOTA for the JSSP instances proposed by [41].
Table A8. Cont.

| ORB6 | MKS   | TDS   | FLT   | ORB7 | MKS   | TDS   | FLT   | ORB8 | MKS   | TDS   | FLT   | ORB9 | MKS   | TDS   | FLT   | ORB10 | MKS   | TDS   | FLT   |
|------|-------|-------|-------|------|-------|-------|-------|------|-------|-------|-------|------|-------|-------|-------|-------|-------|-------|
| 1    | 1090  | *     | 1382.5| 9489 | 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  |       |       |       |       |       |       |

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

| LA01 | MKS   | TDS   | FLT   | LA02 | MKS   | TDS   | FLT   | LA03 | MKS   | TDS   | FLT   | LA04 | MKS   | TDS   | FLT   | LA05 | 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| 4989  | 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  | *    |       |       |       |       |       |       |       |       |       |       |
| Table A9. Cont. |
|----------------|
| LA11 | LA12 | LA13 | LA14 | LA15 |
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1    | 1222 * | 9579.5 | 17,606 | 1039 * | 7550.0 | 14,528 | 1150 * | 8618.0 | 16,397 | 1292 * | 9927.5 | 17,940 |
| 2    | 1234 | 9317.5 | 17,344 | 1045 | 7514.0 | 14,564 | 1150 | 8618.0 | 16,397 | 1292 | 9927.5 | 17,940 |
| 3    | 1238 | 9222.5 * | 17,249 | 1050 | 7498.0 | 14,512 | 1150 | 8618.0 | 16,397 | 1292 | 9927.5 | 17,940 |
| 4    | 1081 | 7318.0 * | 14,332 | 1152 | 8550.0 | 16,397 | 1292 | 9966.0 | 17,940 | 1292 | 9927.5 | 17,940 |
| 5    | 1182 | 8246.0 | 15,577 | 1207 | 9792.5 | 17,940 | 1292 | 9966.0 | 17,940 | 1292 | 9927.5 | 17,940 |
| 6    | 1189 | 7811.0 * | 15,504 | 1217 | 9679.5 | 17,940 | 1292 | 9966.0 | 17,940 | 1292 | 9927.5 | 17,940 |

| LA16 | LA17 | LA18 | LA19 | LA20 |
| 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 |
| 2    | 1008 | 771.0 | 8567 | 830 | 1016.0 | 7782 | 874 | 560.5 | 7836 | 904 | 398.0 | 8071 |
| 3    | 1065 | 613.5 | 8503 | 848 | 1001.0 | 7698 | 905 | 555.0 | 8017 | 916 | 375.0 | 8146 |
| 4    | 1082 | 603.0 | 8227 | 850 | 969.0 | 7569 | 908 | 422.0 | 7880 | 916 | 342.0 | 7903 |
| 5    | 1091 | 490.5 * | 8311 | 854 | 934.5 | 7557 | 922 | 342.0 | 7903 | 916 | 342.0 | 7903 |
| 6    | 1107 | 524.0 | 8130 * | 856 | 8335.5 | 7517 | 933 | 468.5 * | 7824 | 953 | 488.0 | 7759 * |

| LA21 | LA22 | LA23 | LA24 | LA25 |
| 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 |
| 2    | 1172 | 3329.5 | 15,084 | 1050 | 3118.0 | 14,068 | 1118 | 2599.5 | 14,441 | 1052 | 2477.0 | 14,047 |
| 3    | 1174 | 3035.5 | 14,835 | 1053 | 3035.0 | 14,000 | 1158 | 2459.0 | 14,476 | 1054 | 2870.5 | 14,001 |
| 4    | 1177 | 3059.5 | 14,607 | 1070 | 2994.0 | 13,975 | 1160 | 2457.0 | 14,436 | 1060 | 2613.5 | 13,860 |
| 5    | 1202 | 3044.5 | 14,763 | 1079 | 2754.0 | 13,625 | 1160 | 2722.5 | 14,389 | 1070 | 2593.5 | 13,918 |
| 6    | 1204 | 3024.5 | 14,743 | 1081 | 2699.0 * | 13,562 | 1163 | 2437.0 | 14,416 | 1073 | 2598.5 | 13,874 |
| 7    | 1220 | 3032.5 | 14,609 | 1087 | 2761.5 | 14,370 | 1079 | 2547.5 | 13,859 | 1079 | 2547.5 | 13,859 |
| 8    | 1238 | 2881.5 | 14,783 | 1178 | 2408.0 * | 14,384 | 1080 | 2473.0 | 14,063 | 1210 | 2595.5 | 14,373 |
| 9    | 1239 | 2877.5 | 14,666 | 1210 | 2595.5 | 14,373 | 1080 | 2546.5 | 13,858 | 1210 | 2595.5 | 14,373 |
| 10   | 1253 | 2832.5 | 14,696 | 1216 | 2562.5 | 14,340 * | 1087 | 2368.0 * | 13,911 | 1216 | 2562.5 | 14,340 * |

| LA26 | LA27 | LA28 | LA29 | LA30 |
| 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 |
| 2    | 1336 | 7171.5 | 22,944 | 1377 | 7946.0 | 24,194 | 1326 | 7315.0 | 23,338 | 1337 | 8513.0 | 23,286 |

| LA31 |
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1    | 1393 | 7295.0 | 22,489 | 1393 | 7295.0 | 22,489 | 1393 | 7295.0 | 22,489 | 1393 | 7295.0 | 22,489 |
Table A9. Cont.

| LA31 | LA32 | LA33 | LA34 | LA35 |
|------|------|------|------|------|
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1 1784 * | 219,44.5 | 44,731 | 1850 * | 22,413.0 | 47,111 | 1719 * | 22,824.5 | 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 | 44,939 | 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 | 21,944.5 | 44,398 | 1792 | 22,657.5 | 45,669 | 1917 | 23,481.5 | 46,709 |
| 5 1842 | 20,988.5 | 43,775 | 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 | 1896 | 21,958.5 | 46,812 | 1746 | 21,446.5 | 43,900 | 1796 | 22,150.5 | 45,162 | 1923 | 23,368.5 * | 46,596 |
| 7 1897 | 21,509.5 | 46,363 | 1972 | 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 | 1723 | 21,777.5 | 44,190 | 1776 | 22,657.5 | 45,669 | 1917 | 23,481.5 | 46,709 |
| 12 2084 | 21,294.5 * | 46,148 | 1777 | 20,945.5 | 43,399 |
| 13 1602 | 7109.5 * | 23,312 * |

| LA36 | LA37 | LA38 | LA39 | LA40 |
|------|------|------|------|------|
| MKS  | TDS  | FLT  | MKS  | TDS  | FLT  | MKS  | TDS  | FLT  |
| 1 1467 * | 3203.0 | 20,649 | 1652 * | 2898.5 | 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 | 2605.0 | 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 | 2844.5 | 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    | MKS    | TDS  | FLT    |
| 1      | 1296 * | 565.0  | 11,621 | 991 * | 587.5  | 8826   | 991 * | 587.5  | 8826   | 991 * | 587.5  | 8826   | 991 * | 587.5  | 8826   | 991 * | 587.5  |
| 2      | 1306   | 692.5  | 11,581 | 999    | 460.5  | 8658   | 796 * | 3124.0 | 14,127 | 821 * | 3504.0 | 14,883 | 837 * | 3263.0 | 14,378 |
| 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   | 3504.0 | 14,883 | 837   | 3263.0 | 14,378 |
| 6      | 1334   | 407.5  | 11,786 | 1037   | 439.0  | 8674   | 847   | 2817.0 | 13,416 | 823   | 3447.0 | 14,826 | 837   | 3263.0 | 14,378 |

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    | MKS    | TDS  | FLT    | MKS    |
| 1      | 1160 * | 3154.5 | 20,470 | 1155 * | 3592.0 | 21,112 | 1138 * | 2732.5 | 19,941 | 1225 * | 4078.0 | 20,098 |
| 2      | 1166   | 2654.0 | 19,808 | 1159   | 3545.0 | 21,105 | 1154   | 2543.0 | 19,839 | 1228   | 3780.0 | 21,149 |
| 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   | 3068.5 | 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    | MKS    | TDS  | FLT    | MKS    |
| 1      | 1469 * | 2284.0 | 19,027 | 1649 * | 7293.0 | 28,872 | 1649 * | 7293.0 | 28,872 | 1649 * | 7293.0 | 28,872 | 1649 * | 7293.0 | 28,872 |
| 2      | 1502   | 2201.0 | 19,461 | 1655   | 7264.0 | 28,843 | 2098 * | 8414.5 | 38,534 | 2126 * | 21,558.0 | 55,423 |
| 3      | 1515   | 1792.5 | 18,791 | 1672   | 7049.0 | 28,696 | 2103   | 7979.0 | 38,146 | 2127   | 21,553.0 | 55,453 |
| 4      | 1519   | 1783.5 | 18,801 | 1673   | 7045.0 | 28,692 | 2125   | 7247.5 | 37,366 | 2156   | 21,540.0 | 55,405 |
| 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 | 54952.0 |
| 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 * |
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. Science 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.; Martinez, 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. Intel. 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. [CrossRef]

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. Wisitsriphanich, 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.; Rodríguez Bazan, E.D.; Martínez, 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. 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]