Yugoslav Journal of Operations Research
xx (20xx), Number xx, xxx–xxx
DOI: https://doi.org/10.2298/YJOR241215001N

Research Article

TWO-STAGE CHAIN-REENTRANT HYBRID FLOW-SHOP WITH DETERIORATING JOBS

Mohamed NEDJAI
RECITS Laboratory, Faculty of mathematics, USTHB, Algiers, Algeria
mohamed161994@yahoo.fr, ORCID: 0009-0004-5069-8440

Karim AMROUCHE
National School of Nanoscience and Nanotechnology, Sidi Abdellah, Algiers, Algeria
k.amrouche@ensnn.dz, ORCID: 0000-0002-0347-2244

Mourad BOUDHAR*
RECITS Laboratory, Faculty of mathematics, USTHB, Algiers, Algeria
mboudhar@usthb.dz, ORCID: 0000-0002-1624-9340

Received: December 2024 / Accepted: January 2026

Abstract: In this paper, we study a two-stage chain-reentrant hybrid flow shop with deteriorating jobs as follows. Each job must initially be scheduled on a primary machine M_1 (first stage) which is then scheduled on one of a set of m unrelated parallel machines (second stage) and returns back to M_1 for its last operation. The jobs are subjected to a linear deterioration function of their starting times. The aim is to minimize both of the makespan and the total energy consumption. For the resolution of this problem, we have developed a mixed-integer linear programming model. We then implement and compare two metaheuristics: a nondominated sorting-based multiobjective genetic algorithm (NSGA2) and an archived multiobjective simulated annealing algorithm (AMOSA). The experimental study indicates that AMOSA generally outperforms NSGA2 on small instances (for all tested values of m and $n \leq 30$), whereas NSGA2 yields better performance on larger instances (for all tested values of m and $n \geq 100$), with respect to both quantity and quality measures. For small instances, the comparison with the exact method (i.e., the mathematical model) confirms that the reference fronts generated by both algorithms are close

*Corresponding author

to the true Pareto front. Finally, we have proposed a TOPSIS-based method to select a representative solution from the Pareto set according to the decision-maker's preferences.

Keywords: Scheduling, deteriorating jobs, bi-objective, NSGA2, AMOSA, TOPSIS.

MSC: 90B35, 90C29, 68W50.

1. INTRODUCTION

1.1. Context and motivations

The minimization of the energy consumption has become a crucial topic in various industrial and technological fields. The increasing demand in energy and environmental concerns have pushed researchers, engineers, and decision-makers to explore innovative strategies to reduce energy consumption while improving systems performance [1]. This trend is particularly relevant in manufacturing, computer science, transportation and resource management sectors [2, 3].

Within contemporary industrial settings, there is a growing need to simultaneously minimize the total duration required to complete all jobs (makespan) and total energy consumption (TEC). Concrete examples of this bi-objective problem can be encountered in the following fields:

- Logistics and Transportation: Planning routes for the delivery of goods minimizing both transportation time (makespan) and vehicle fuel consumption (TEC).
- Industrial Production: Scheduling production lines to minimize the total time needed to complete all jobs (makespan) and equipment usage (TEC) simultaneously.

Due to the explosion of industrial competition, companies seek to attract customers, increase their market share and ensure their profitability. This has increasingly led to the appearance of new constraints such as chain-reentrant and deterioration which we recall their meanings as follows. In the usual flow shop each job visits each machine only once, but there are some real-world situations where a job can visit a machine or a set of machines more than once. Such problems are called reentrant-shops and can be found in semiconductor manufacturing, where each component may revisit the same machine several times. As regards the deterioration, the processing times vary over time unlike in the traditional models where the processing times are fixed. In such problems, the processing time can be an increasing or a decreasing function of the starting time in the schedule. An example of such an increasing function includes the firefighting, where any delay in activating the fire suppression protocol (intervention protocol) increases the complexity and the time required to fully extinguish the fires.

Our problem focuses on minimizing both the makespan and the total energy consumption, considering the effects of chain reentrance and deterioration in a hybrid flow shop system. This scheduling problem is motivated by a real-world industrial application, particularly in the field of metallurgy, such as steel rolling mills. In such context, each metal piece (job) typically follows a two-stage processing route. First, it is treated on a primary rolling machine M_1 (first stage). Then, it is sent to one of several parallel heating or treatment units (second stage), which differ in processing characteristics and this justifies the

modeling of the second stage using unrelated parallel machines. Finally, the job returns to the same rolling machine M_1 for a final finishing operation, resulting in a chain-reentrant structure. Moreover, delays between stages often cause temperature drops in the material, which in turn increase the rolling time due to reduced malleability. This phenomenon is naturally captured by a linear deterioration model where processing times depend on the job's starting time. In parallel, the energy consumption of machines increases with both processing time and delay, especially when additional heating or reactivation is required. These practical aspects are embedded into our formulation via a deterioration-dependent energy model. Consequently, our problem provides a realistic framework that reflects the essential trade-off between productivity and energy efficiency in modern industrial systems.

1.2. Contributions

The most popular models in industry are flowshop-type workshops. In fact, numerous contemporary industrial applications require the incorporation of both chain-reentrant and deterioration concepts and these have been taken into account in our work. Since there is a lack of literature combining the chain-reentrant and deterioration constraints while minimizing the makespan and the total energy consumption simultaneously, the present paper treats these constraints in a two-stage hybrid flow-shop system.

The remainder of the paper is organized as follows: the next section presents a literature review on the chain-reentrant flow shop problem with the aim of minimizing makespan and the total energy consumption, and deteriorating jobs. In Section 3, we present a detailed description of the problem in its general form, and a Mixed-Integer Linear Programming model. Section 4 is devoted to the resolution methods, in which we propose two meta-heuristics: NSGA2 and AMOSA. In Section 5, we first have selected the parameters' values of the previous meta-heuristics by using ANOVA analysis and have then compared them empirically using three measures: quantity, quality and average time. The empirical comparisons have proved that AMOSA is better for small instances, whereas NSGA2 performs well in large instances for the first two measures. AMOSA is considerably faster when considering the average time measure. By implementing our MILP model, we have validated the performance of the proposed metaheuristics on small-size instances. We also propose a TOPSIS-based method to find the best solution from the approximated reference front, in order to meet the decision-maker' preferences. We conclude in Section 6 this document by some perspectives for our future work.

2. LITERATURE REVIEW

The classical flow-shop scheduling problems assume that the jobs pass through all machines in the same order, and that each job must visit every machine only once. However, in reality this assumption is not always justified. This has lead [4]. to introduce the chain-reentrant flow shop problem, where a job can visit a certain machine or a set of machines more than once.

Later, many other researchers have considered various models of such problems which are summarized as follows. [5] investigated the model where each job must previously be treated on the primary machine, then on a certain number of machines in the same order

and returns back to the primary machine to achieve its last operation. The aim is to minimize the makespan. The authors have proved some properties on the optimal solutions, in the case of two machines. [6] studied the two machine reentrant flow shop scheduling problem with the objective of minimizing the total tardiness. In this model, all the jobs must be processed twice on each machine: each job should be processed on machine 1, machine 2 and then machine 1 and machine 2. The authors developed dominance properties, a lower bound and heuristic algorithms for the problem, and used them to develop a branch and bound algorithm. [7] have considered a polynomial sub-case of the problem of Wang et al. previously cited. The authors have given the optimal solution for the makespan minimization. They have also studied the same chain-reentrant flow shop problem of [5], but by considering the minimization of the total flow time. The same authors [7] have divided this minimization problem into sub-problems according to a relation between the execution times and have proposed algorithms and some dominant properties for their resolutions. [8] addressed a problem called reentrant hybrid flow shop with serial stages, where each stage consists of identical parallel machines and that a job may revisit any stage several times. The objective is to minimize simultaneously the makespan and total tardiness. They proposed local-search based Pareto genetic algorithms to approximate the Pareto optimal solutions. The same authors then compared these algorithms with the existing multi-objective genetic algorithm NSGA2, in terms of the convergence to optimal solution, the diversity and the dominance of solution.

[9] considered a specific hybrid reentrant flow shop scheduling problem. Their shop contains some stages with some identical parallel machines. They developed an exact method which lists all the solutions and selects the best one and have also devised approximated methods such as the genetic algorithm (GA), the genetic algorithm under fuzzy controller (FLCGA), the particle swarm optimization (PSO) and the particle swarm optimization under fuzzy controller (FLCPSO). [10] studied a reentrant parallel machines scheduling problem with consumable resources. Each job consumes several components and must be processed more than once in a stage composed of identical parallel machines. Since this problem is NP-hard, they elaborated a MILP model in order to find the exact solution and a genetic algorithm to obtain near-optimal solution for large instances. Then, an improvement phase based on different local search procedures were performed and examined to generate better solutions. The experimental results showed that the presented algorithm is able to find an optimal solution for small-sized instances and can effectively find a near optimal solution for large-sized instances to minimize the makespan of the considered problem. [11] investigated deeply another model of the chain-reentrant flow-shop, where each job must be scheduled on the sequence M_1,M_2 and returns back to M_1 for its last operation. In this problem an exact time lag between the two operations on the first machine is imposed and the goal is to minimize the makespan. These authors have proposed some heuristics for the problem in the the case of identical time lags and have established a new NP-hard result and some polynomials cases. Other studies on reentrant shop scheduling can be found in [12], [13], [14], [15] and [16].

The multi-objective optimization has witnessed a growing interest and a significant development in many academic and industrial fields. For the literature in this field, one can find the paper of [17]. These authors studied the distributed flexible job-shop scheduling problem (DFJSP) for sand casting production. They proposed a DFJSP model. A hy-

brid teaching–learning-based optimization (HTLBO) algorithm is developed to solve this model. Experimental results show that their approach outperforms six other algorithms. Another paper is [18] where the authors considered the Multiobjective Distributed No-Idle Permutation Flowshop Scheduling problem, aiming to minimize the makespan and total tardiness simultaneously. They proposed an Iterative Greedy Algorithm with a Q-learning mechanism and demonstrated that this algorithm provided more satisfactory results. We also cite the paper [19] in which the authors studied the many-objective sand casting whole process production scheduling problem. They constructed a multi-stage hybrid flowshop scheduling model to minimize makespan, machine load, penalty of tardiness and earliness, and carbon emission. They then proposed a discrete group teaching optimization algorithm (DGTOA) to solve the model. They proved that the DGTOA maintained a good balance between exploration and development.

The expansion of multi-objective optimization has led researchers to consider various constraints, particularly in reentrant shop problems with multiple criteria. Among the notable results published in this area, we cite the following papers. [20] considered a reentrant hybrid flow shop scheduling problem with two objectives: the maximization of the utilization rate of the bottleneck and the minimization of the maximum completion time. For the resolution, the authors provided an exact method and three meta-heuristics for which two well-known algorithms: the Non-dominated Sorting Genetic Algorithm version 2 (NSGA 2) of [21] and the Strength Pareto Evolutionary algorithm version2 (SPEA2) of [22]. The third is a new one called Lorenz Non-dominated Sorting Genetic Algorithm (L-NSGA). By using two standard multi-objective metrics, they concluded that L-NSGA provided better solutions than NSGA2 and SPEA2 and that its solutions are closer to the optimal front. The same authors addressed another model of multiobjective reentrant hybrid flow shop [23]. The shop is composed of several stages made of several identical parallel machines. The problem is to minimize two objectives: the makespan and the total tardiness of the tasks. They improved a new method with different local searches: Adjacent and Non Adjacent Pairwise Interchange, Extract and Backward-Shifted Reinsertion, and Extract and Forward-Shifted Reinsertion. Every local search is tuned with statistical method (design of experiment) and the best one is worked out. They compared this method with the best one in several instances. In [24], the authors have extended the problem proposed by [11] by considering the bi-objective problem minimizing the total energy consumption cost and the makespan. They proposed a bi-objective mixed integer programming formulation. An ε-constraint method and NSGA2 approach were then been proposed to obtain the optimal Pareto front and approximate Pareto solutions for the problem.

In recent years, the energy consumption has become a subject of great importance in manufacturing, which prompted researchers to produce a vast amount of literature with the aim of developing various strategies for reducing energy usage. [25] considered the distributed energy-efficient parallel machines scheduling problem (DEPMSP) minimizing both total energy consumption and total tardiness. They proposed a knowledge-based two-population optimization (KTPO) algorithm and compared it with four algorithms from the literature. Their comparative results and statistical analysis proved the effectiveness and advantages of KTPO in solving DEPMSP. In [26], the authors used a Q-learning-based hyperheuristic (HHQL) to address the energy-efficient distributed blocking flow

shop scheduling problem (EEDBFSP). They developed an initialization method considering total tardiness and energy consumption. By the experimental results, the authors showed that HHQL outperforms other algorithms.

To the best of our knowledge, We have noticed that there is still a lack of the published literature that treats the reentrant flow shop minimizing both the makespan and the total energy consumption.Motivated by this, we initiated this research with an abstract submitted to the ORSSA 2021 conference [27] and have further developed it into the present paper. In this paper, we address a bi-objective scheduling problem, which we have called: the two-stage chain-reentrant hybrid flow-shop with deteriorating jobs. This constraint of deterioration will be defined afterwards. The shop of our problem is composed of two stages. The first contains a single machine M_1 called the primary machine whereas the second contains m unrelated parallel machines. Each job must initially be scheduled on M_1 (first stage) which is then scheduled on one of the m unrelated parallel machines of the second stage and returns back to M_1 for its last operation. The aim is to minimize both of the makespan (C_{max}) and the total energy consumption (TEC).

As far as the jobs deterioration is concerned, we suppose that the job processing times are an increasing function of their starting times and this terminates the definition of our problem. The phenomenon of jobs deterioration has extensively been studied in various machine settings and performance measures. For a review on deterioration , the reader can see the papers ([28], [29], [30], [31], [32] and [33]).

To summarize the existing contributions and clarify the distinction between the different types of reentrant flow shop scheduling problems found in the literature, Table 1 presents a comparative view of the most relevant works. The classification is based on shop configuration, objective(s), solution method(s), and key contributions.

Table 1: Summary of related work on reentrant flow shop problems

Reference	**Notation**	**Objectives**	**Methods / Contributions**
Graves et al. (1983)	Chain-R-FSP	$C_{\max}$	Structural results, poly. cases
Choi & Kim (2009)	2M-R-FSP	$\sum T_j$	Dominance rules, B&B, heuristics
Chu et al. (2010)	Chain-R-FSP	$\sum C_j$	Polynomial subcases, heuristics
Cho et al. (2011)	RHFSP-Serial	$C_{\max}, \sum T_j$	GA, NSGA2 comparison
Yalaoui et al. (2014)	RHFSP-Parallel	$C_{\max}$	Exact, GA, PSO, Fuzzy control
Belkaid et al. (2016)	R-FSP + Resources	$C_{\max}$	MILP, GA, local search
Amrouche et al. (2016)	Chain-R-FSP + TL	$C_{\max}$ + TL	Complexity, heuristics
Dugardin et al. (2010)	MO-RHFSP	$C_{\max}$, Util., $\sum T_j$	NSGA2, SPEA2, local search
Liu et al. (2017)	BiO-R-FSP + Energy	$C_{\max}$, Energy	MILP, NSGA2, ε-constraint
Pan et al. (2022)	EEDBFSP / DEPMSP	TEC, $\sum T_j$	KTPO, Q-learning HH
This paper	Chain-RHFSP + Deterioration	$C_{\max}$, TEC	MILP, NSGA2, AMOSA, TOPSIS

3. PROBLEM DESCRIPTION AND MATHEMATICAL MODEL

3.1. Problem description

The two-stage chain-reentrant hybrid flow-shop with deteriorating jobs problem consists in scheduling a set of n jobs $J_1, J_2, ..., J_n$ on a set of $m+1$ machines: M_1 (called the primary machine) and m unrelated parallel machines $M_2, M_3, ..., M_{m+1}$. Each job must

first be treated on M_1 (first stage), then on one of the m unrelated parallel machines (second stage) and returns back to M_1 for its last operation.

Before presenting the deterioration constraints and the problem formulation, we start by introducing the complete set of parameters, inputs, and some useful notations.
- Inputs:

- n : number of jobs
- $T = \{J_1, J_2, \ldots, J_n\}$: set of jobs
- m : number of unrelated parallel machines of the second stage
- $U = \{M_2, \ldots, M_{m+1}\}$: set of unrelated parallel machines in stage 2
- $p_{i1}^{0-(1)}, p_{i1}^{0-(2)}$: initial processing times of job J_i on M_1 ($1^{\text{st}}$and 2^{nd} passage respectively)
- p_{ij}^{0} : initial processing time of job ($J_i \in T$) on machine ($M_j \in U$)

- Parameters:

- $s_{i1}^{(1)}, s_{i1}^{(2)} \geq 0$: start times of job J_i at first and second passage on M_1
- $s_{ij} \geq 0$: start time of job $J_i \in T$ on machine $M_j \in U$
- $p_{i1}^{(1)}, p_{i1}^{(2)}, p_{ij}$: processing times
- $C_i \geq 0$: completion time of job $J_i \in T$
- $C_{\max} \geq 0$: makespan

The shop involves deterioration constraints which are described as follows:
We suppose that the processing times: $p_{i1}^{(1)}$, $p_{i1}^{(2)}$ (first and second passage on M_1) and p_{ij} (second stage) for each job J_i are subjected to linear deterioration constraints which are given by the following formulae:

- On M_1 (first passage): $p_{i1}^{(1)} = p_{i1}^{0-(1)} + \alpha_{i1} s_{i1}^{(1)}$.
- On M_j ($j \in \overline{2, m+1}$): $p_{ij} = p_{ij}^{0} + \alpha_{ij} s_{ij}$.
- On M_1 (second passage): $p_{i1}^{(2)} = p_{i1}^{0-(2)} + \alpha_{i1} s_{i1}^{(2)}$.

where the value α_{ij} is an input representing the deterioration rate of the job J_i on the machine M_j.

Our problem is bi-objective and consists in minimizing both the makespan (C_{max}) and the total energy consumption (TEC) which are defined as follow:

The completion time C_i of job J_i is $C_i = s_{i1}^{(2)} + p_{i1}^{(2)}$. Since the jobs are processed without interruption, the makespan $C_{\max}$ is defined as the maximum of all job completion times, i.e., $C_{\max} = \max_{i=1}^{n} C_i$.

In order to define the total energy consumption (TEC), we also suppose that in our shop, each machine M_j consumes during its treatment an amount of energy $\delta_j \geq 0$ per unit

of time. Therefore, the energies consumed in M_1 for the treatment of the job J_i are $p_{i1}^{(1)}\ \delta_1$ (first passage), $p_{i1}^{(2)}\ \delta_1$ (second passage) and in M_j is $p_{ij}\ \delta_j$ (unrelated parallel machines), and all of these energies are increased by rates $\theta_j \geq 0$ over time. Then, the total energies consumed to process J_i on M_j are:

- On M_1: $E_{i1}^{(1)} = p_{i1}^{(1)}\delta_1 + \theta_1 s_{i1}^{(1)}$ and $E_{i1}^{(2)} = p_{i1}^{(2)}\delta_1 + \theta_1 s_{i1}^{(2)}$.
- On M_j: $E_{ij} = p_{ij}\delta_j + \theta_j s_{ij}$.

The total energy consumed is:

$$TEC = \sum_{J_i \in T} \left(E_{i1}^{(1)} + E_{i1}^{(2)} + \sum_{M_j \in U} E_{ij}\right).$$

A schedule of jobs is said to be feasible, if it respects the chain-reentrant passages order $M_1 \rightarrow M_j \in U \rightarrow M_1$. Our bi-objective problem consists in finding a feasible schedule that minimizes both of the makespan (C_{max}) and the total energy consumption (TEC) simultaneously.

Using the three-field notation of [34], our problem is denoted by
(P): $FH2(1^{(1)}, R_m^{(2)})|ChR, p_{ij}^0 + \alpha_{ij}s_{ij}|C_{max}, TEC$.

We now present an illustration of a feasible schedule of our problem (P).

Example:

Consider the problem with 4 jobs J_1, J_2, J_3, J_4 and 3 machines: M_1 is the primary machine, M_2, M_3 are the unrelated parallel machines. The rest of inputs are represented on Table 2.

- n: the number of jobs $(n = 4)$
- m: the number of unrelated parallel machines $(m = 2)$
- $p_{i1}^{0-(1)}$ and $p_{i1}^{0-(2)}$: the initial processing times of the job J_i corresponding to the first and the second passages on M_1 respectively
- p_{ij}^0: the initial processing time of the job J_i on the machine M_j $(j \neq 1)$
- α_{ij}: the deterioration rate of the job J_i on the machine M_j
- δ_j: the energy consumption of the machine M_j per unit of time
- θ_j: the rate of energy increase of the machine M_j

Table 2: Inputs of the Example

α_{ij}	M_1	M_2	M_3
J_1	0.2	0.87	0.51
J_2	0.1	0.7	0.77
J_3	0.15	0.71	0.89
J_4	0.25	0.95	0.17

J_i	$p_{i1}^{0-(1)}$	p_{i2}^0	p_{i3}^0	$p_{i1}^{0-(2)}$
J_1	1	4	6	3
J_2	2	10	8	1
J_3	1	8	6	2
J_4	3	3	3	1

M_j	M_1	M_2	M_3
δ_j	0.31	0.6	0.45
θ_j	0.25	0.1	0.2

A feasible solution of the problem (P) is represented by the Gantt diagram on Figure 1 with $C_{max} = 29.42$ and $TEC = 45.81$.

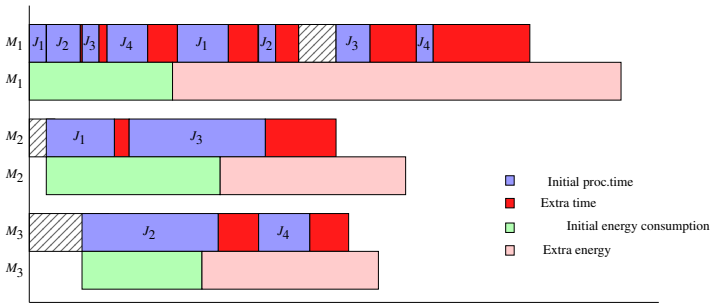

Figure 1: Feasible solution of the problem P

We define three sub-problems of (P) noted (P_1), (P_2) and (P_3) related to the deterioration constraints and the energy consumption on M_1. These sub-problems are represented on Table 3.

Table 3: List of sub-problems

Sub-problems		The corresponding constraints
(P_1)	• The deteriorating ratios are independent of the machines M_j and are strictly positive • Both of the energy consumption and the increasing ratios are strictly positive	$\alpha_{ij} = \alpha_i > 0, i \in \{1,n\}, j \in \{1,m+1\}$ $\delta_j > 0, \theta_j > 0, j \in \{1,m+1\}$
(P_2)	• The deteriorating ratios are independent of the machines M_j $(j \neq 1)$ and are strictly positive • The deterioration is not taken into consideration on M_1 • Both of the energy consumption and the increasing ratios are strictly positive	$\alpha_{ij} = \alpha_i > 0, i \in \{1,n\}, j \in \{2,m+1\}$ $\alpha_{i1} = 0, i \in \{1,n\}$ $\delta_j > 0, \theta_j > 0, j \in \{1,m+1\}$
(P_3)	• The deteriorating ratios are independent of the machines M_j $(j \neq 1)$ and are strictly positive • The deterioration is not taken into consideration on M_1 • Both of the energy consumption and the increasing ratios are strictly positive except for M_1 • Both of the energy consumption and the increasing ratios are not taken into consideration on M_1	$\alpha_{ij} = \alpha_i > 0, i \in \{1,n\}, j \in \{2,m+1\}$ $\alpha_{i1} = 0, i \in \{1,n\}$ $\delta_j > 0, \theta_j > 0, j \in \{2,m+1\}$ $\delta_1 = \theta_1 = 0$

As we have already presented a real-world application of our problem (P), note that the two sub-problems (P_2) and (P_3) can be encountered in the real world when the machine M_1 is a mean of transport. The transport of the metallic pieces can be done by an automatic trolley as in the case of the sub-problem (P_2) or by means of a manual trolley as in the case of the sub-problem (P_3).

3.2. Mathematical model

In order to complete our model, further notations and parameters are added as follows.
- Further notations:

- $A = \{M_1, M_2, \ldots, M_{m+1}\}$: set of all machines (including M_1)

- Let $J_{i1}^{(1)}, J_{i1}^{(2)}$ denote the two operations of job $J_i \in T$ corresponding to the first and second passages on M_1 respectively. Let T' be the set of the $2n$ operations defined by: $T' = \{J_{i1}^{(1)}, J_{i1}^{(2)} \mid J_i \in T\}$
- Let $L \in T'$ be an operation on M_1. L is either $J_{i1}^{(1)}$ or $J_{i1}^{(2)}$ for a job $J_i \in T$. We denote by s_L and p_L respectively the start time and processing time of operation L.
- M : a sufficiently large constant (big-M)

- Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if job } J_i \text{ is assigned to machine } M_j \in U \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ikj} = \begin{cases} 1 & \text{if job } J_i \text{ is scheduled before job } J_k \text{ on machine } M_j \in U \\ 0 & \text{otherwise} \end{cases}$$

$$order_{LZ} = \begin{cases} 1 & \text{if operation } L \text{ is scheduled before operation } Z \text{ on machine } M_1,\ L,Z \in T' \\ 0 & \text{otherwise} \end{cases}$$

Two objective functions are considered in the studied problem, expressed as follows:

$$\min C_{\max} \tag{1}$$
$$\min TEC \tag{2}$$

As regards the total energy consumption function, we have:

$$TEC = \sum_{J_i \in T} (E_{i1}^{(1)} + E_{i1}^{(2)} + x_{ij} \sum_{M_j \in U} E_{ij}).$$

$$TEC = \sum_{J_i \in T} (p_{i1}^{(1)} \cdot \delta_1 + \theta_1 \cdot s_{i1}^{(1)} + p_{i1}^{(2)} \cdot \delta_1 + \theta_1 \cdot s_{i1}^{(2)} + x_{ij} \sum_{M_j \in U} (p_{ij} \cdot \delta_j + \theta_j \cdot s_{ij})).$$

Notice that the total energy consumption function is non-linear due to the products $x_{ij} \cdot s_{ij}$ and $x_{ij} \cdot p_{ij}$. To linearize these terms, we introduce two auxiliary variables:

$$t_{ij}^s = x_{ij} \cdot s_{ij} \quad (t_{ij}^s \geq 0) \quad \text{and} \quad t_{ij}^p = x_{ij} \cdot p_{ij} \quad (t_{ij}^p \geq 0).$$

Consequently, we obtain the linear form of TEC as follows:

$$TEC = \sum_{J_i \in T} (p_{i1}^{(1)} \cdot \delta_1 + \theta_1 \cdot s_{i1}^{(1)} + p_{i1}^{(2)} \cdot \delta_1 + \theta_1 \cdot s_{i1}^{(2)} + \sum_{M_j \in U} (t_{ij}^p \cdot \delta_j + \theta_j \cdot t_{ij}^s))$$

along with the big-M linearization constraints, cited below in the MILP-constraints.

- MILP constraints:
- Assignment constraints:

$$\sum_{M_j \in U} x_{ij} = 1, \qquad \forall J_i \in T \tag{3}$$

- Routing constraints(Reentrance):

$$s_{ij} \geq s_{i1}^{(1)} + p_{i1}^{(1)} - M(1-x_{ij}), \qquad \forall J_i \in T,\ \forall M_j \in U \tag{4}$$

$$s_{i1}^{(2)} \geq s_{ij} + p_{ij} - M(1-x_{ij}), \qquad \forall J_i \in T,\ \forall M_j \in U \tag{5}$$

- Completion constraints:

$$C_i = s_{i1}^{(2)} + p_{i1}^{(2)}, \qquad \forall J_i \in T \tag{6}$$

$$C_{\max} \geq C_i, \qquad \forall J_i \in T \tag{7}$$

- The big-M linearization constraints (energy calculation):

$$t_{ij}^{s} \leq s_{ij}, \quad t_{ij}^{s} \leq Mx_{ij}, \quad t_{ij}^{s} \geq s_{ij} - M(1-x_{ij}), \qquad \forall J_i \in T,\ \forall M_j \in U \tag{8}$$

$$t_{ij}^{p} \leq p_{ij}, \quad t_{ij}^{p} \leq Mx_{ij}, \quad t_{ij}^{p} \geq p_{ij} - M(1-x_{ij}), \qquad \forall J_i \in T,\ \forall M_j \in U \tag{9}$$

- Non-overlapping constraints on M_1:

$$s_L + p_L \leq s_Z + M(1-order_{LZ}), \qquad \forall L,Z \in T_M,\ L \neq Z \tag{10}$$

$$s_Z + p_Z \leq s_L + Morder_{LZ}, \qquad \forall L,Z \in T_M,\ L \neq Z \tag{11}$$

$$order_{LZ} + order_{ZL} = 1, \qquad \forall L,Z \in T_M,\ L \neq Z \tag{12}$$

- Non-overlapping constraints on stage 2:

$$s_{ij} + p_{ij} \leq s_{kj} + M(1-y_{ikj}), \qquad \forall J_i,J_k \in T,\ i \neq k,\ \forall M_j \in U \tag{13}$$

$$s_{kj} + p_{kj} \leq s_{ij} + My_{ikj}, \qquad \forall J_i,J_k \in T,\ i \neq k,\ \forall M_j \in U \tag{14}$$

$$y_{ikj} + y_{kij} = 1, \qquad \forall J_i,J_k \in T,\ i \neq k,\ \forall M_j \in U \tag{15}$$

Objective 1 is to minimize the makespan. Objective 2 is to minimize the total energy consumption. Constraint (3) ensures that each job is assigned to exactly one machine in stage 2. Constraints (4) and (5) model the chain-reentrance routing of jobs: each job starts its second stage after the first passage on M_1 and returns again to M_1 for its second passage. Constraints (6) and (7) define the job completion time and the makespan. Constraints (8) and (9) represent the big-M linearization constraints. Constraints (10), (11), and (12) ensure non-overlapping between operations on M_1, meaning that only one operation in first or second passage can be processed at a time on M_1. Constraints (13), (14), and (15) represent the non-overlapping constraints in stage 2. They ensure that no two jobs assigned to the same machine $M_j \in U$ are processed simultaneously.

4. SOLUTION APPROACHES

In this section, we propose a list algorithm-based heuristic and two multiobjective meta-heuristics for the resolution of our problem (P). The first one is the Non Dominated Sorting Genetic Algorithm (NSGA2) [35] and the second is the Archived Multi-Objective Simulated Annealing (AMOSA) [36]. Moreover, we present a decision-making method (TOPSIS) (technique for order of preference by similarity to ideal solution) which determines the selection of the best solution among the solutions obtained from the multiobjective approaches.

4.1. The encoding

The encoding is a step which translates a scheduling solution into a chromosome or vice-versa. For our problem a chromosome is represented by a vector whose size is the number of jobs. A position in a chromosome (gene) represents a job number (i.e job-based representation).

Figure 2 represents the encoding of a feasible solution. The job J_5 is scheduled on the first position on the machine M_1, followed by the job J_7, and so on until the job J_6 which is scheduled on the last position. This encoding is adopted for the implementation of the proposed heuristic as well as for the two meta-heuristics NSGA2 and AMOSA.

5	7	4	2	1	8	3	6

Figure 2: Encoding of a solution

4.2. Solution construction and evaluation

Each solution of our problem is constructed and then evaluated by a list algorithm based on giving a chromosome, this algorithm is called LS-Algorithm defined below.

Algorithm 1 LS-Algorithm

1: Schedule the jobs on the primary machine M_1 according to a given priority list.
2: Among the m unrelated parallel machines, choose the least loaded one and schedule the first available job on this machine.
3: Schedule again the jobs on M_1 according to the FCFS rule (First come first served).

4.3. NSGA2

The Non-dominated Sorting Genetic Algorithm (NSGA2), introduced by [21], is a widely recognized multi-objective optimization algorithm known for its efficiency in handling complex problems with conflicting objectives. Its key innovation lies in simultaneously optimizing multiple objectives while maintaining diversity among solutions. NSGA2 improves upon traditional genetic algorithms by incorporating non-dominated sorting and crowding distance mechanisms, ensuring a balance between convergence (finding optimal solutions) and diversity (exploring the solution space), which helps avoid premature convergence to local optima.

In our work, NSGA2 is employed to minimize both makespan (C_{max}) and total energy consumption (TEC). The implementation process involves:

Genetic representation: We begin by generating an initial population of individuals (chromosomes).

Population generation: New generations are created using crossover and mutation genetic operators applied to the current population.

Fitness evaluation: For each solution (chromosome), we calculate its fitness: The values of C_{max} and TEC.

Selection: To determine which chromosomes are more likely to achieve the best results, a selection is made. In our algorithm we used two key principles for this:

- **Non-domination (Pareto sense):** A solution i dominates another solution j if $F_1(i) \leq F_1(j)$ and $F_2(i) \leq F_2(j)$, where F_1 and F_2 correspond to C_{max} and TEC, respectively. The population is sorted accordingly, and non-dominated solutions form the Pareto front.
- **Crowding distance:** If solutions belong to the same Pareto front, we calculate a crowding distance ($dist_i$) to maintain diversity. The crowding distance is computed based on the objective function values $f_1(i)$ and $f_2(i)$ of each solution, normalized using the maximum and minimum values of each objective in the current population.

The **crowding distance algorithm** follows these steps [37]:

Algorithm 2 Crowding Distance Method

Initialize: N solutions on the Pareto front.
for $i = 1$ to N
  $dist_i = 0$
for $k = 1$ to 2
  $SORT(f,k)$
  $dist_1 = dist_N = \infty$
  for $i = 2$ to $N-1$
    $dist_i = dist_i + \frac{f_{i+1}^k - f_{i-1}^k}{f_k^{max} - f_k^{min}}$

The NSGA2 algorithm proceeds as follows: We generate an initial population $\mathbf{P}_0$ of size l, evaluate the fitness of each individual, and then select individuals based on non-domination and crowding distance to form a set of non-dominated solutions. These solutions undergo crossover and mutation operations to create a new population $\mathbf{C}$ with improved potential. The combined population $\mathbf{P} \cup \mathbf{C}$ (size $2l$) is evaluated again, and the selection process is repeated to maintain non-dominated individuals. This iterative process continues until a stopping criterion is met. Figure 3 illustrates the general workflow of the NSGA2 algorithm [35].

Finally, we give the pseudo-code of the NSGA2 in Algorithm 3.

4.4. AMOSA

The Archived Multi-objective Simulated Annealing (AMOSA) is a sophisticated metaheuristic designed to handle multi-objective optimization problems, as introduced by [36]. AMOSA extends the traditional simulated annealing algorithm by incorporating the concept of an archive, which stores the non-dominated solutions identified during the search process. This archive serves as a crucial innovation, allowing the algorithm to efficiently manage and maintain a diverse set of Pareto-optimal solutions throughout the optimization process. One of the primary advantages of AMOSA is its ability to balance exploration and exploitation, and this is a critical aspect of multi-objective optimization. By integrating the archive notion, AMOSA ensures that the solutions found do not only focus on a single objective (which could neglect other objectives), but that they are also well distributed to represent a diversity of possible choices. This prevents the algorithm to be

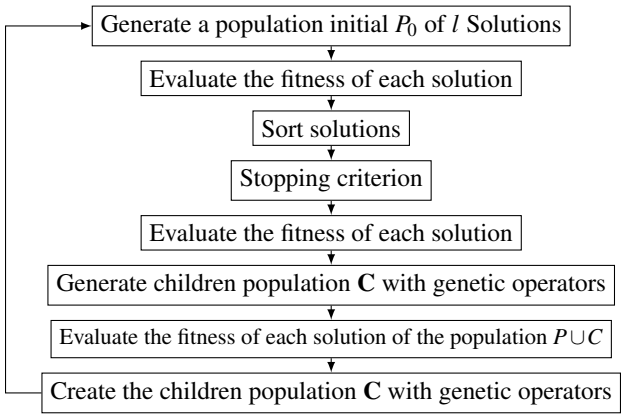

Figure 3: General operating scheme of the NSGA2 algorithm

Algorithm 3 NSGA2 Algorithm pseudo-code

Initialize: l, t, $Itermax$, $Pcros$, $Pmut$.
Generate a random initial population P_0 of size l.
$t = 1$
Evaluate the fitness function of each solution "$fitness = C_{max}$ and TEC".
Sort solutions by **non-domination** and **crowding distance** to obtain the non-dominated groups.
while $Itermax > 1$
    Create the children population C of size l:
    Select two parents from the population.
    Generate probability P_1 and P_2.
    if $(P_1 \leq P\_crossover)$
        Crossover of the parents.
    else
        Copy of the parents.
    if $(P_2 \leq P\_mutation)$
        Mutation of the children.
    Evaluate the fitness function of each solution of the population $P \cup C$.
    Sort solutions by **non-domination** and **crowding distance** to obtain the non-dominated groups.
    $t = t + 1$
    $Itermax = Itermax - 1$
Return the non-dominated groups.

focused on a single objective to the detriment of others. This characteristic is particularly beneficial in complex industrial applications where multiple conflicting objectives must be considered simultaneously.

AMOSA begins with an initial solution and gradually explores the solution space through the simulated annealing process, where solutions are iteratively improved based on a probabilistic acceptance criterion. The algorithm's archive plays a key role in storing non-dominated solutions as they are discovered. The size of this archive is kept bounded,

as only a limited number of well-distributed Pareto-optimal solutions are necessary. This ensures that the search remains focused and efficient while covering the entire Pareto front. AMOSA's ability to maintain a diverse and high-quality set of solutions makes it particularly suitable for problems where a comprehensive exploration of trade-offs between objectives is required.

The steps of the algorithm are detailed as follows:

Archive initialization: The algorithm starts by initializing a number of initial solutions in the archive. A solution is accepted into the archive if and only if it is not dominated by any other solution from the archive already stored.

Domination degree: AMOSA uses the concept of domination degree to calculate the probability of acceptance of a degrading solution. This degree of domination is defined as follows: given two solutions a and b, the degree of domination $\delta dom_{a,b}$ is calculated by the function $\delta dom_{a,b} = \prod_{i=1,f_i(a)\neq f_i(b)}^{M} \frac{|f_i(a)-f_i(b)|}{R_i}$ where M is the number of objectives (M=2) and R_i represents the variation range of the $i-th$ objective function.

At the start of the algorithm, one of the points called "current solution" is randomly selected from the initial archive, at a temperature $temp = Tmax$. The current solution *current_sol* is then perturbed to generate a new solution *new_sol*. The domination state of *new_sol* is checked with respect to the *current_sol* and to the solutions in the archive. Based on the state of domination between *current_sol* and *new_sol*, three cases can be distinguished:

1. **Case 1**: *current_sol* dominates *new_sol*, and k solutions in the archive dominate $new-sol$ ($k \geq 1$). In this case, *new_sol* will be accepted with a probability $prob = \frac{1}{1+e^{\delta dom_{avg} temp}}$ $(\diamond)$

   with $\delta dom_{avg} = \frac{\sum_{i=1}^{k} \delta dom_{i,new\_sol} + \delta dom_{current\_sol,new\_sol}}{k+1}$

   where δdom_{avg} represents the average degree of domination of the points *new_sol* and the $k+1$ solutions which are respectively the *current_sol* solution and the k solutions of the archive.

   According to the equation $(\diamond)$, the two parameters that control the probability value are δdom_{avg} and the temperature $temp$. The greater these two values, the less is the probability of accepting the dominated solution *new_sol*. This is explained by the principle of the $temp$ parameter and its diminution throughout the course of the iterations of the algorithm. Thus, the further we advance in the search, the probability of accepting dominated solutions also decreases. Moreover, the parameter δdom_{avg} takes a significant value if the quality of the dominated solution *new_sol* is worse compared to the non-dominated solutions of the archive, and consequently the probability of accepting this solution decreases.

2. **Case 2**: *current_sol* and *new_sol* are mutually non dominated. In this case, two sub-cases may arise:
   a- *new_sol* is also not dominated by any other solution in the archive, then *new_sol* belongs to the solution front. Therefore, *new_sol* is added to the archive, and assigned to *current_sol*. Archive solutions that are dominated by *new_sol* will obviously be eliminated.

b- *new_sol* is dominated by k solutions from the archive $(k > 0)$, then we are in a case similar to case (1) and so *new_sol* will be accepted with a probability calculated according to the equation $(\diamond)$ previously presented, except that in this case, δdom_{avg} is calculated differently: $\delta dom_{avg} = \frac{\sum_{i=1}^{k} \delta dom_{i,new\_sol}}{k}$

3. **Case 3**: *new_sol* dominates *current_sol*: in this case, two possibilities may arise:
   a- $new - sol$ is dominated by k solutions from the archive $(k \geq 1)$. This case is only possible if a dominated solution has been accepted and assigned to current-sol. Here, we take the minimum of $\delta dom_{i,new\_sol}$, where $i = 1...k$. This means that the solution i^* of the archive which has the minimum degree of dominance with respect to *new_sol* will then be assigned to *current_sol* with a probability equal to $frac11 + e^{-\delta dom_{i^*}}$.

   b- *new_sol* is not dominated by any solution of the archive. In this situation, *new_sol* belongs to the front of the archive and it is added there, and $current\_sol = new\_sol$. Obviously, the solutions in the archive dominated by *new_sol* will be eliminated.

   The process is repeated $nb\_iteration$ times for each temperature $temp$. The temperature is reduced to $\alpha \times temp$, by using the cooling rate of α until the minimum temperature **Tmin** is reached and the search process stops. Finally, the archive contains the final non-dominated solutions forming the Pareto optimal front.

Lastly, we give the pseudo-code of AMOSA in Algorithm 4.

Algorithm 4 AMOSA Algorithm

```
Initialize (Archive, variables, current_sol)
temp = Tmax
while temp > Tmin
    for i = 0 to iter
        new_sol = perturb(current_sol)
        if current_sol dominates new_sol                                  case 1
            new_sol = current_sol with probability
        if current_sol and new_sol are non-dominating to each other       case 2
            if new_sol is dominated by k solutions in the Archive         sub-case 2.1
                new_sol = current_sol with probability
            if new_sol is non-dominating by any solutions from the Archive  sub-case 2.2
                new_sol = current_sol; add new_sol to the Archive and remove all
                solutions dominated by new_sol from the Archive
        if new_sol dominates current_sol                                  case 3
            if new_sol is dominated by k solutions in the Archive         sub-case 3.1
                current_sol = solution with the minimum degree of domination
                from the Archive with probability
            if new_sol is non-dominating by any solutions from the Archive  sub-case 3.2
                new-sol = current_sol; add new_sol to the Archive and remove all
                solutions dominated by new_sol from the Archive
    temp = α × temp                                          update the temperature
return Archive.
```

4.5. TOPSIS

The Technique for Order Performance by Similarity to Ideal Solution (TOPSIS) is a widely recognized multi-objective decision-making technique, introduced by [38]. It is particularly effective in ranking and selecting the best solutions from a set of Pareto-optimal solutions. The core innovation of TOPSIS lies in its ability to quantify the trade-offs between conflicting objectives by measuring The Euclidean distance of each solution from an ideal solution (the best possible outcome) and an anti-ideal solution (the worst-case scenario). TOPSIS is especially useful in scenarios where decision-makers need to make informed choices based on multiple criteria. By providing a clear and quantifiable ranking of the solutions, it helps in identifying the most balanced and desirable outcomes. This technique is well-suited to complex industrial applications where multiple conflicting objectives need to be optimized simultaneously, offering a straightforward yet powerful method for decision-making.

The principle of TOPSIS is based on the distance between the alternatives compared to the positive ideal V^+ (called ideal point) and the negative ideal V^- (called anti-ideal). The positive ideal V^+ represents a fictitious solution taking the best value (among the values of the alternatives) of each criterion. Conversely, the negative ideal V^- takes the worst values.

The TOPSIS method is defined in seven steps:

1. Establish a matrix X, where a set of alternatives $a_1, a_2, ..., a_i, ..., a_k$ are compared to criteria $c_1, c_2, ..., c_j, ..., c_n$.
2. Normalize the decision matrix to obtain a new matrix $R = (r_{ij})$ such that:
   $r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{n} x_{ij}^2}}$
3. Calculate the weighted normalized matrix $V = (v_{ij})$ such that $v_{ij} = w_j r_{ij}$. Note that the weights w_j are given by the decision makers in order to represent their preferences between the criteria, with $\sum_{j=1}^{n} w_j = 1$.
4. Define the positive ideal V^+ and the negative ideal V^-:
   $V^+ = \{max_i\{v_{ij}, j \in j^+\}, min_i\{v_{ij}, j \in j^-\}\} \equiv \{v_j^+, j = 1, ..., n\}$.
   $V^- = \{min_i\{v_{ij}, j \in j^+\}, max_i\{v_{ij}, j \in j^-\}\} \equiv \{v_j^-, j = 1, ..., n\}$.
   where
   j^+ is the set of criteria to be maximized, j^- is the set of criteria to be minimized.
5. Calculate for each alternative, the euclidean distances between the positive ideal and the negative ideal, denoted d_j^+ and d_j^- respectively.
   $d_i^+ = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^+)^2}$ and $d_i^- = \sqrt{\sum_{j=1}^{n} (v_{ij} - v_j^-)^2}$
6. Calculate the proximity degree to the positive ideal D_i^+. The larger D_i^+, the closer the alternative is to the positive ideal and the farthest from the negative ideal: $D_i^+ = \frac{d_i^-}{d_i^- + d_i^+}$
7. Finally, sort the solutions with respect to D_i^+. The alternatives will be sorted by preference order.

5. NUMERICAL EXPERIMENTS

5.1. Preliminary tests

The parameters of the two meta-heuristics are chosen by generating a set of thirty randomly generated instances for different values of $n = \{10, 30, 50, 100, 200\}$ and $m = \{2, 5, 10, 20\}$. For this parameters' selection we have conducted simulations and have evaluated the results by using an ANOVA analysis at a significance level of 5%. The value of a parameter is chosen according to the following protocol. We have generated all the possible combinations for the parameters' values of each meta-heuristic. Several tests have then been realized on the previous set of thirty instances by fixing the values of one parameter and leaving the others non fixed. The considered and selected values for the parameters of NSGA2 and AMOSA are given in Tables 4 and 5.

Table 4: Preliminary tests NSGA2

NSGA2			
Problem	Parameters	Tested values	Selected values
(P)	Number of iteration	600, 800, 1000	1000
	Size of initial population	40, 60, 80	60
	P_crossover	0.2, 0.5, 0.8	0.5
	P_mutation	0.2, 0.5, 0.8	0.2
(P_1)	Number of iteration	600, 800, 1000	1000
	Size of initial population	40, 60, 80	80
	P_crossover	0.2, 0.5, 0.8	0.8
	P_mutation	0.2, 0.5, 0.8	0.2
(P_2)	Number of iteration	600, 800, 1000	600
	Size of initial population	40, 60, 80	40
	P_crossover	0.2, 0.5, 0.8	0.5
	P_mutation	0.2, 0.5, 0.8	0.5
(P_3)	Number of iteration	600, 800, 1000	600
	Size of initial population	40, 60, 80	40
	P_crossover	0.2, 0.5, 0.8	0.5
	P_mutation	0.2, 0.5, 0.8	0.2

5.2. Experimental results

In this part, we have implemented the two meta-heuristics NSGA2 and AMOSA proposed in section 3, for the resolution of the problems (P), (P_1), (P_2) and (P_3). The implementation is done on a Pentium *i*3 PC 2.00 GHZ and 4.00 GB Ram with the C programming language. All the instances have randomly been generated and are classified into four classes as follows:

- The first class corresponds to $m = 2$ with 5 values of n, $n \in \{10, 30, 50, 100, 200\}$.
- The second class corresponds to $m = 5$ with 5 values of n, $n \in \{10, 30, 50, 100, 200\}$.
- The third class corresponds to $m = 10$ with 5 values of n, $n \in \{10, 30, 50, 100, 200\}$.
- The fourth class corresponds to $m = 20$ with 5 values of n, $n \in \{10, 30, 50, 100, 200\}$.

The instances of each class are constructed as follows: for each pair (m, n), we have generated 4 sets of 100 instances corresponding to problems (P), (P_1), (P_2) et (P_3). Then the total number of the generated instances is equal to $4 \times (5 \times (100 \times 4)) = 8000$.

Table 5: Preliminary tests AMOSA

AMOSA			
Problem	Parameters	Tested values	Selected values
(P)	Size of Archive	80, 100, 120	100
	Number of repetitions	5, 7, 10	10
	Cooling_coefficients	0.75, 0.85, 0.95	0.95
	Initial temperature	0.1, 0.3, 0.5	0.5
	Final temperature	10^{-5}, 10^{-4}, 10^{-3}	10^{-5}
(P_1)	Size of Archive	80, 100, 120	80
	Number of repetitions	5, 7, 10	10
	Cooling_coefficients	0.75, 0.85, 0.95	0.95
	Initial temperature	0.1, 0.3, 0.5	0.3
	Final temperature	10^{-5}, 10^{-4}, 10^{-3}	10^{-5}
(P_2)	Size of Archive	80, 100, 120	120
	Number of repetitions	5, 7, 10	10
	Cooling_coefficients	0.75, 0.85, 0.95	0.95
	Initial temperature	0.1, 0.3, 0.5	0.3
	Final temperature	10^{-5}, 10^{-4}, 10^{-3}	10^{-5}
(P_3)	Size of Archive	80, 100, 120	80
	Number of repetitions	5, 7, 10	10
	Cooling_coefficients	0.75, 0.85, 0.95	0.95
	Initial temperature	0.1, 0.3, 0.5	0.5
	Final temperature	10^{-5}, 10^{-4}, 10^{-3}	10^{-5}

It is clear that evaluating the effectiveness of a meta-heuristic in the multi-criteria case is not as simple as in the single-criteria case. [39] introduced different measures for evaluating the effectiveness of heuristics and multi-criteria meta-heuristics. We mainly distinguish measures that provide information concerning the comparison between an approximation and the best known approximation.

To evaluate the performances of these two meta-heuristics and compare them empirically, we have considered three criteria for each. The first and second criteria are the quantity and quality of the approximation calculated by the two algorithms. Let ZE_N and ZE_A be the two Pareto optimal fronts generated by the NSGA2 and AMOSA algorithms respectively, and let ZE^* be the Pareto optimal front of $ZE_N \cup ZE_A$. Thus, the best Pareto optimal front is ZE^* because it contains the non-dominated solutions which belong to $ZE_N \cup ZE_A$. We call it the reference front. The third criterion is the time measure. These three measures are described as follows:

Quantity measure: Let $Q_1(Z)$ be the percentage of the non-dominated solutions in Z which are also in ZE^* with respect to the reference front ZE^*. We have,

$$Q_1(Z) = 100 \times \frac{|ZE^* \cap Z|}{|ZE^*|}$$

For each pair (n,m), we calculate the average value of $Q_1(ZE_N)$ and $Q_1(ZE_A)$.

Quality measure: Let $Q_2(Z)$ be the percentage of non-dominated solutions in Z that are also in ZE^* with respect to the front Z. We have

$$Q_2(Z) = 100 \times \frac{|ZE^* \cap Z|}{|Z|}$$

For each pair (n,m), we calculate the average value of $Q_2(ZE_N)$ and $Q_2(ZE_A)$.
Time measure: The third criterion is the average time denoted t_{avg}, in seconds for each meta-heuristic. After an extensive experimental comparison of meta-heuristics NSGA2 and AMOSA, the experimental results obtained for the four classes of instances are presented by Tables 6, 7, 8 and 9. In each of these tables, the first row corresponds to $Q_1(Z)$ which represents the average value of the quantity measure in percentage, while the second row corresponds to $Q_2(Z)$ represents the average value of the quality measure. The last line (t_{avg}) corresponds to the average CPU time in seconds.

Tests for $m = 2$

Table 6: Experimental results for $m = 2$

	problem P		sub problem P_1		sub problem P_2		sub problem P_3	
	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA
n=10								
$Q_1(Z)$	18.77	81.23	40.5	59.5	16	84	16.67	83.33
$Q_2(Z)$	19.1	89.36	40.5	81.15	1.6	84	1.8	84
t_{avg}	0.91	0.25	1.48	0.19	0.32	0.04	0.29	0.05
n=30								
$Q_1(Z)$	27.06	72.94	61.29	38.71	8	92	7	93
$Q_2(Z)$	34.2	83.27	61.29	44.42	0.8	92	0.7	93
t_{avg}	4.25	0.74	5.23	0.61	1.7	0.12	1.65	0.13
n=50								
$Q_1(Z)$	58.57	41.43	92.19	7.81	22.33	77.67	26.03	73.97
$Q_2(Z)$	76.59	53.31	92.19	7.99	3.5	78.11	3.2	74.1
t_{avg}	13.61	1.77	16.91	1.62	6.17	0.79	5.68	0.68
n=100								
$Q_1(Z)$	79.57	20.43	78.39	21.61	89.65	10.35	86.63	10.37
$Q_2(Z)$	99.19	27.6	99.4	29.54	49.4	14.5	45.8	17.78
t_{avg}	86.64	7.72	106.74	7.17	44.64	4.82	37.37	4.06
n=200								
$Q_1(Z)$	86.5	13.5	85.79	14.21	99.9	0.1	98.72	1.28
$Q_2(Z)$	99.9	16.5	99.9	17.36	92.5	0.2	84.59	2.4
t_{avg}	623.64	41.07	395.3	39.09	274.44	13	269.98	16.85

Observations and analysis case $m = 2$:
1/ For the quantity $Q_1(Z)$ and quality $Q_2(Z)$, we observe that :

- AMOSA dominates NSGA2 when $(n \leq 30)$ whereas NSGA2 is better than AMOSA for $(n \geq 100)$, whatever the problems P, P_1, P_2 and P_3 are.
- For the value $n = 50$, NSGA2 performs better than AMOSA in the case of the problems P and P_1 however AMOSA is better when considering the problems P_2 and P_3.

2/ As for the average time t_{avg}, AMOSA is considerably fast, in effect the highest value observed is $t_{avg} = 41.07$ seconds.

Tests for $m = 5$

Observations and analysis case $m = 5$:

Table 7: Experimental results for $m = 5$

	problem P		sub problem P_1		sub problem P_2		sub problem P_3	
	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA
n=10								
$Q_1(Z)$	19.59	80.41	21.84	78.16	8	92	16	84
$Q_2(Z)$	20.4	89.85	21.6	96.31	0.8	92	16	84
t_{avg}	0.9	0.23	1.53	0.19	0.31	0.05	0.29	0.04
n=30								
$Q_1(Z)$	29.68	70.32	29.97	70.03	3	97	3	97
$Q_2(Z)$	35.5	79.54	35.79	79.7	0.3	97	0.3	97
t_{avg}	4.17	0.75	5.46	0.62	1.78	0.11	1.62	0.11
n=50								
$Q_1(Z)$	61.43	38.57	68.6	31.4	6	94	4	96
$Q_2(Z)$	76.8	49.7	86.59	41.79	0.6	94	0.4	96
t_{avg}	13.64	1.74	17.26	1.55	6.15	0.27	5.63	0.26
n=100								
$Q_1(Z)$	79.82	20.18	79.5	20.5	53	47	37	63
$Q_2(Z)$	98.59	27.16	98.7	27.05	5.3	47	3.7	63
t_{avg}	86.34	7.96	106.69	7.16	38.49	1.35	36.61	1.3
n=200								
$Q_1(Z)$	86.38	13.62	85.26	14.74	98	2	97	3
$Q_2(Z)$	99.8	16.7	100	18.66	9.8	2	9.7	3
t_{avg}	621.05	42.12	389.99	37.21	276.53	9.14	270.37	9.07

1/ For the quantity $Q_1(Z)$ and quality $Q_2(Z)$, we observe that:

- AMOSA dominates NSGA2 when $(n \leq 30)$ whereas NSGA2 is better than AMOSA for $(n = 200)$, whatever the problems P, P_1, P_2 and P_3 are.
- For the value $n = 50$, NSGA2 performs better than AMOSA in the case of the problems P and P_1 however AMOSA is better when considering the problems P_2 and P_3.
- For the value $n = 100$, NSGA2 performs better than AMOSA in the case of the problems P and P_1 however AMOSA is better when considering the problem P_3. For the problem P_2, we observe a slight dominance of NSGA2 for the quantity measure $Q_1(Z)$ whereas AMOSA is better for the quality measure $Q_2(Z)$.

2/ As for the average time t_{avg}, AMOSA is considerably fast, in effect the highest value observed is $t_{avg} = 42.12$ seconds.

Tests for $m = 10$
Observations and analysis case m=10:

1/ For the quantity $Q_1(Z)$ and quality $Q_2(Z)$, we observe that:

- AMOSA dominates NSGA2 when $(n \leq 30)$ whereas NSGA2 is better than AMOSA for $(n = 200)$, whatever the problems P, P_1, P_2 and P_3 are.
- For the value $(n = 50, n = 100)$, NSGA2 performs better than AMOSA in the case of the problems P and P_1 even though AMOSA is better when considering the problems P_2 and P_3.

Table 8: Experimental results for $m = 10$

	problem P		sub problem P_1		sub problem P_2		sub problem P_3	
	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA
n=10								
$Q_1(Z)$	21.91	78.09	29.18	70.82	5,5	94.5	9	91
$Q_2(Z)$	26.29	89.67	34.09	85.26	0.6	95	1	92
t_{avg}	0.92	0.26	1.5	0.22	0.29	0.05	0.29	0.04
n=30								
$Q_1(Z)$	30.93	69,07	32.05	67.95	4	96	5	95
$Q_2(Z)$	38.59	80,25	39.4	78.27	0,4	96	0.5	95
t_{avg}	4.25	0,78	5.58	0.66	1.82	0.11	1.65	0.1
n=50								
$Q_1(Z)$	69.99	34.01	66.9	33.1	5	95	5	95
$Q_2(Z)$	83.59	44.61	84.9	43.48	0.5	95	0.5	95
t_{avg}	13.89	1.8	17.46	1.59	6.18	0.26	5.74	0.25
n=100								
$Q_1(Z)$	80.67	19.33	79.74	20.26	53	47	47	53
$Q_2(Z)$	97.69	25.72	99.4	27.67	5.3	47	4.7	53
t_{avg}	86.86	7.7	108.45	7.2	38.95	1.35	36.98	1.31
n=200								
$Q_1(Z)$	87.47	12.53	86.2	13.8	97	3	89	11
$Q_2(Z)$	100	15.18	99.9	16.73	9.7	3	8.9	11
t_{avg}	623.07	42.6	388.75	39.28	272.7	8.98	267.47	9.04

2/ As for the average time t_{avg}, AMOSA is considerably fast, in effect the highest value observed is $t_{avg} = 42.6$ seconds.

Tests for $m = 20$

Table 9: Experimental results for $m = 20$

	problem P		sub problem P_1		sub problem P_2		sub problem P_3	
	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA	NSGA2	AMOSA
n=10								
$Q_1(Z)$	20.08	79.92	26,84	73,16	7	93	4	96
$Q_2(Z)$	23,1	90,46	31,6	86,25	0,8	94	0,6	98
t_{avg}	0,99	0,25	1,6	0,19	0,34	0,04	0,32	0,04
n=30								
$Q_1(Z)$	39,63	60,37	42,7	57,3	1	99	1	99
$Q_2(Z)$	49.29	72.4	51.5	67.73	0.1	99	0.1	99
t_{avg}	5.44	0.98	5.89	0.69	1.94	0.11	1.79	0.11
n=50								
$Q_1(Z)$	65.91	34.09	68.4	31.6	3	97	4	96
$Q_2(Z)$	84.9	45.08	86.09	42.05	0.3	97	0.4	96
t_{avg}	14.23	1.89	17.86	1.66	6.35	0.26	6.02	0.25
n=100								
$Q_1(Z)$	78.15	21.85	78.18	21.82	54	46	34	66
$Q_2(Z)$	98.3	30.47	98	29.2	5.4	46	3.4	66
t_{avg}	87.94	7.7	109.15	7.27	39.23	1.34	37.74	1.34
n=200								
$Q_1(Z)$	85.04	14.96	86.31	13.69	99	1	83	17
$Q_2(Z)$	100	18.73	100	16.91	9.9	1	8.3	17
t_{avg}	618.52	38.32	394.88	38.39	275.06	9.06	276.36	9.31

Observations and analysis case $m = 20$:

1/ For the quantity $Q_1(Z)$ and quality $Q_2(Z)$, we observe that:

- AMOSA dominates NSGA2 when $(n \leq 30)$ whereas NSGA2 is better than AMOSA for $(n = 200)$, whatever the problems P, P_1, P_2 and P_3 are.
- For the value $(n = 50, n = 100)$, NSGA2 performs better than AMOSA in the case of the problems P and P_1 even though AMOSA is better when considering the problems P_2 and P_3.

2/ As for the average time t_{avg}, AMOSA is considerably fast, in effect the highest value observed is $t_{avg} = 38.32$ seconds.

General observations:

The combinations of the previous test results have shown that in general AMOSA dominates NSGA2 for small instances, while NSGA2 dominates AMOSA for large instances and that for the two measures quantity and quality. As regards the average time, AMOSA is considerably fast.

Comparison with the exact model

To validate the effectiveness of our metaheuristics, we have implemented the proposed MILP model with CPLEX 12.8 and have applied it to a set of small-sized instances, due to the complexity of the problem. Specifically, we have generated 15 random instances for each of the following configurations: $m = 2$, $n = 5$ and $m = 2$, $n = 6$. For each instance, the exact Pareto front has been obtained using a weighted-sum method by varying ε in the aggregated objective function $Z_\varepsilon = \varepsilon.C_{max} + (1-\varepsilon).TEC$, with $\varepsilon \in [0,1]$.

We have then compared the approximated reference front obtained from the concatenation of non dominated solutions of the two proposed metaheuristics NSGA2 and AMOSA to the exact Pareto front obtained by the MILP model. To assess the quality of this approximated reference front, we have used standard performance indicators: the Generational Distance (GD), the Inverted Generational Distance (IGD), and the Hypervolume Coverage (HV Coverage). Table 10 reports the average results across all instances for each problem size.

Table 10: Average performance metrics across 15 instances for each configuration

Instance size	Mean GD	Mean IGD	Mean HV Coverage
$m = 2, n = 5$	2.6068	2.5076	142.72%
$m = 2, n = 6$	3.3657	2.8971	189.45%

Globally, the values of the performance indicators confirm that the approximated reference fronts obtained from concatenation of non dominated solutions of the two proposed metaheuristics are close to those given by the exact Pareto fronts. The HV Coverage, in particular, have shown promising results, reflecting good convergence and diversity of the approximated reference fronts.

To illustrate the quality of the approximated reference fronts, Figures 4 and 5 present a graphical comparison with the exact ones for two representative instances, $(m = 2, n = 5)$ and $(m = 2, n = 6)$.

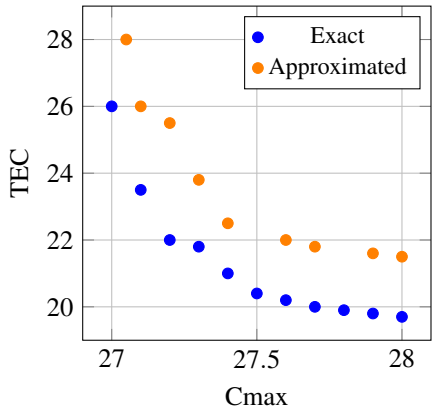

Figure 4: Instance $m = 2, n = 5$

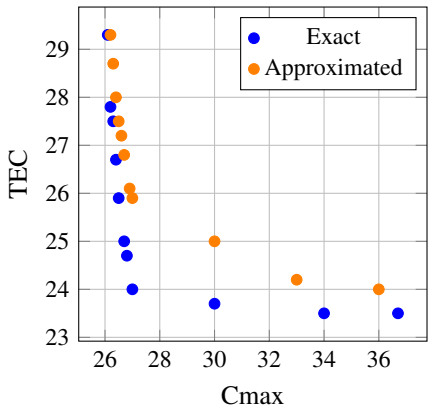

Figure 5: Instance $m = 2, n = 6$

5.3. TOPSIS-Based method

In order to meet the decision-maker's requirements, we propose a TOPSIS-Based method in order to choose the best solutions for different weight combinations. First of all, we give an example illustrating our TOPSIS-Based method by considering a reference front of an instance $m = 10, n = 10$, composed of 10 non dominated solutions (points), obtained from concatenation of non dominated solutions of the two proposed methods NSGA2 and AMOSA. Each of these solutions is presented through its evaluation of the makespan and total energy consumption as shown in Table 11.

Table 11: Reference front instance $m = 10, n = 10$

Sol_i	Sol_1	Sol_2	Sol_3	Sol_4	Sol_5	Sol_6	Sol_7	Sol_8	Sol_9	Sol_{10}
C_{max}	56.12	56.76	55.74	56.59	54.98	59.03	57.78	55.10	57.32	55.23
TEC	70.47	67.44	72.94	68.38	91.48	66.31	66.55	81.42	66.89	76.59

After applying our TOPSIS-Based method to this reference front with weight combinations $(0.7, 0.3)$ and $(0.3, 0.7)$, where the first and the second weights of each combination correspond to the makespan and the total energy consumption respectively, we have obtained Figures 6 and 7. The red points are those generated by NSGA2 and the blue points correspond to AMOSA.

Figure 6 corresponds to the combination $(0.7, 0.3)$, where the best solution by our TOPSIS-Based method is $Sol_4 = (56.59, 68.38)$ which is indicated by the red encircled point. As for the combination $(0.3, 0.7)$, the best solution by our TOPSIS-Based method is $Sol_9 = (57.32, 66.89)$ and is indicated by the blue encircled point on Figure 7.

By applying our TOPSIS-Based method to this reference front for a big range of representative combination weights, we obtain an order of solutions from best to worst as indicated in Table 12.

After conducting additional experiments by generating well distributed instances and applying our TOPSIS-Based method for three representative combinations, we have ob-

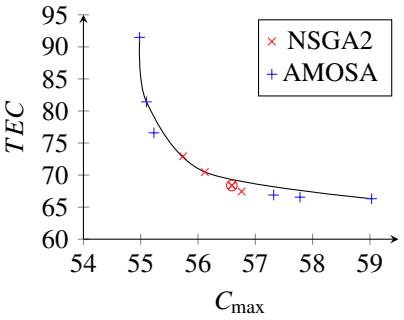

Figure 6: The best solution for combination $(0.7, 0.3)$

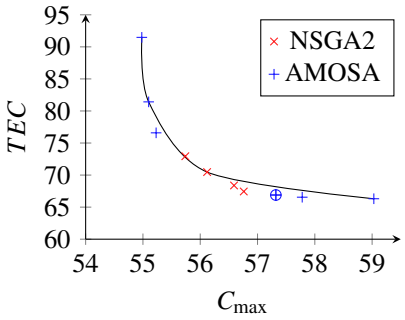

Figure 7: The best solution for combination $(0.3, 0.7)$

tained the results displayed on Table 13 containing the best and worst solutions for each instance.

The experimental results show that even when the data is perturbed, our method produces accurate and reliable solutions. In addition, it manages to do so in extremely short deadlines, proving therefore both the robustness and effectiveness of our approach.

Table 12: TOPSIS-Based results for different combinations

Weights	**Obtained Order**
$(0.9, 0.1)$	$10-3-8-1-4-5-2-9-7-6$
$(0.8, 0.2)$	$1-3-4-2-10-9-7-8-6-5$
$(0.7, 0.3)$	$4-2-1-9-3-7-6-10-8-5$
$(0.6, 0.4)$	$2-4-9-1-7-6-3-10-8-5$
$(0.5, 0.5)$	$2-9-4-7-1-6-3-10-8-5$
$(0.4, 0.6)$	$2-9-7-4-6-1-3-10-8-5$
$(0.3, 0.7)$	$9-2-7-6-4-1-3-10-8-5$
$(0.2, 0.8)$	$7-9-6-2-4-1-3-10-8-5$
$(0.1, 0.9)$	$7-6-9-2-4-1-3-10-8-5$

6. CONCLUSION

In this paper, we have studied a novel bi-objective problem called the two-stage chain reentrant hybrid flow shop problem with deteriorating jobs. The objective is to find the best schedule minimizing both of the makespan and the total energy consumption.

For the resolution of the problem, we have proposed and empirically compared two metaheuristics: NSGA2 and AMOSA, from which we have derived an approximated reference front. We have then proposed a TOPSIS-based method to determine the best solution from this approximated reference front, in order to meet the decision-maker's preferences. In addition, we have proposed a new exact MILP model which has enabled us to validate the performance of the proposed metaheuristics on small-size instances.

Table 13: TOPSIS-Based results for different instances

Instances	Combination weights (makespan-total energy consumption)					
	0.7 – 0.3		0.5 – 0.5		0.3 – 0.7	
	Best	Worst	Best	Worst	Best	Worst
$m = 2, n = 10$	58.72 – 116.21	61.11 – 114.01	58.79 – 115.98	57.77 – 128.97	58.79 – 115.98	57.77 – 128.97
$m = 2, n = 30$	394.32 – 475.14	456.09 – 453	401.17 – 469.63	456.09 – 453	412.16 – 464.8	389.9 – 501.68
$m = 2, n = 50$	2386.15 – 2758.17	2556.65 – 2560.88	2488.91 – 2609.39	2371.04 – 2862.5	2530.6 – 2578.87	2371.04 – 2862.5
$m = 2, n = 100$	187208 – 360331	176755 – 520131	188326 – 357366	176755 – 520131	200206 – 323180	176755 – 520131
$m = 5, n = 10$	64.2 – 75.34	66.92 – 74.35	64.4 – 74.98	64.09 – 81.18	64.4 – 74.98	64.09 – 81.18
$m = 5, n = 30$	441.15 – 323.89	486.82 – 310.83	441.15 – 323.89	432.41 – 349.42	464.06 – 314.6	432.41 – 349.42
$m = 5, n = 50$	1704.53 – 4678	1846.43 – 4518.58	1749.41 – 4554.84	1846.43 – 4518.58	1749.41 – 4554.84	1704.53 – 4678
$m = 5, n = 100$	151391 – 281902	131887 – 310254	131887 – 310254	126247 – 396763	142370 – 286982	126247 – 396763
$m = 10, n = 10$	47.87 – 35.21	47.2 – 43.42	48.19 – 34.46	47.2 – 43.42	48.19 – 34.46	47.2 – 43.42
$m = 10, n = 30$	467.83 – 362.55	497.95 – 354.6	467.83 – 362.55	493.42 – 359.25	497.95 – 354.6	466.85 – 376.24
$m = 10, n = 50$	2068.39 – 3307.57	2285.76 – 3060.45	2168.8 – 3136.25	2285.76 – 3060.45	2168.8 – 3136.25	2045.01 – 3405.9
$m = 10, n = 100$	177832 – 383063	173416 – 465535	182319 – 363293	173416 – 465535	195271 – 340882	173416 – 465535
$m = 20, n = 10$	55.13 – 66.38	56.53 – 66.47	55.13 – 66.38	54.51 – 71.04	55.23 – 66.09	54.51 – 71.04
$m = 20, n = 30$	405.47 – 473.25	412.26 – 471.33	413.74 – 462.05	412.26 – 471.33	413.74 – 462.05	403.79 – 479.34
$m = 20, n = 50$	1613.55 – 593.25	1935.95 – 576.82	1613.55 – 593.25	1935.95 – 576.82	1657.15 – 581.26	1935.95 – 576.82
$m = 20, n = 100$	224437 – 502625	212287 – 680318	230493 – 479837	212287 – 680318	236558 – 465492	212287 – 680318

As perspectives, further research could explore the application and comparison of other advanced meta-heuristics, such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). In addition, the implementation of exact methods (e.g., branch-and-bound or dynamic programming). Another possible extension of this work is to investigate two-agent optimization scenarios, considering either shared or distinct objectives.

Acknowledgements: The authors gratefully wish to thank the anonymous reviewers for their careful reading of this paper and for their valuable and useful comments. Their contributions helped to improve this paper.

Funding: This research received no external funding.

REFERENCES

[1] A. Pechmann and I. Schöler, "Optimizing energy costs by intelligent production scheduling," in *Glocalized Solutions for Sustainability in Manufacturing*. Springer, Berlin, Heidelberg, 2011, p. 293–298. ISBN 978-3-642-19691-1

[2] G. Kant and K. Sangwan, "Predictive modeling for power consumption in machining using artificial intelligence techniques," *Procedia CIRP*, vol. 26, p. 403–407, 2015. doi: 10.1016/j.procir.2014.07.072

[3] L. Shu, Y. He, and T. Hu, "An on-line approach for energy efficiency monitoring of machine tools," *Journal of Cleaner Production*, vol. 27, p. 133–140, 2012. doi: 10.1016/j.jclepro.2012.01.013

[4] S. Graves, H. Meal, D. Stefek, and A. Hamid, "Scheduling of re-entrant flow shops," *Journal of Operations Management*, vol. 3, no. 4, p. 197–207, 1983. doi: 10.1016/0272-6963(83)90004-9

[5] M. Y. Wang, S. P. Sethi, and S. L. van de Velde, "Minimizing makespan in a class of reentrant shops," *Operations Research*, vol. 45, no. 5, p. 702–712, 1997. doi: 10.1287/opre.45.5.702

[6] S.-W. Choi and Y.-D. Kim, "Minimizing total tardiness on a two machine reentrant flowshop," *European Journal of Operational Research*, vol. 199, p. 375–384, 2009. doi: 10.1016/j.ejor.2008.11.037

[7] F. Chu, C. Chu, and C. Desprez, "Series production in a basic reentrant shop to minimize makespan or total flow time," *Computers & Industrial Engineering*, vol. 58, no. 2, p. 257–268, 2010. doi: 10.1016/j.cie.2009.02.017

[8] H.-M. Cho, S.-J. Bae, J. Kim, and J. I.-J., "Bi-objective scheduling for reentrant hybrid flow shop using pareto genetic algorithm," *Computers & Industrial Engineering*, vol. 61, no. 3, p. 529–541, 2011. doi: 10.1016/j.cie.2011.04.008

[9] N. Yalaoui, L. Amodeo, F. Yalaoui, and H. Mahdi, "Efficient method to schedule reentrant flowshop system," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 26, no. 3, p. 1113–1121, 2014. doi: 10.5555/2596417.2596422

[10] F. Belkaid, F. Yalaoui, and Z. Sari, "An efficient approach for the reentrant parallel machines scheduling problem under consumable resources constraints," *International Journal of Information Systems and Supply Chain Management*, vol. 9, no. 3, p. 1–25, 2016. [Online]. Available: https://ideas.repec.org/a/igg/jisscm/v9y2016i3p1-25.html

[11] K. Amrouche, M. Boudhar, M. Bendraouche, and F. Yalaoui, "Chain-reentrant shop with an exact time lag: new results," *International Journal of Production Research*, vol. 55, p. 1–11, 2016. doi: 10.1080/00207543.2016.1205235

[12] C. H. Pan and J. S. Chen, "Mixed binary integer programming formulations for the reentrant job shop scheduling problem," *Computers & Operations Research*, vol. 32, no. 5, p. 1197–1212, 2005. doi: 10.1016/j.cor.2003.10.004

[13] C. Jing, W. Huang, and G. Tang, "Minimizing total completion time for re-entrant flow shop scheduling problems," *Theoretical Computer Science*, vol. 412, no. 48, p. 6712–6719, 2011. doi: 10.1016/j.tcs.2011.08.030

[14] K.-C. Ying, S.-W. Lin, and S.-Y. Wan, "Bi-objective reentrant hybrid flowshop scheduling: an iterated pareto greedy algorithm," *International Journal of Production Research*, vol. 52, no. 19, p. 5735–5747, 2014. doi: 10.1080/00207543.2014.910627

[15] J. Shen, L. Wang, J. Deng, and X. Zheng, "A pareto-based discrete harmony search algorithm for bi-objective reentrant hybrid flowshop scheduling problem," in *Harmony Search Algorithm*, 2016, vol. 382, p. 435–445. ISBN 978-3-662-47925-4

[16] A. P. Rifai, H. T. Nguyen, and S. Z. M. Dawal, "Multiobjective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Applied Soft Computing*, vol. 40, p. 42–57, 2016. doi: 10.1016/j.asoc.2015.11.034

[17] H. Tang, B. Fang, R. Liu, Y. Li, and S. Guo, "A hybrid teaching and learning-based optimization algorithm for distributed sand casting job-shop scheduling problem," *Applied Soft Computing*, vol. 120, p. 108694, 2022. doi: 10.1016/j.asoc.2022.108694

[18] F. Zhao, C. Zhuang, L. Wang, and C. Dong, "An iterative greedy algorithm with q-learning mechanism for the multiobjective distributed no-idle permutation flowshop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 5, p. 3207–3219, 2024. doi: 10.1109/TSMC.2024.3358383

[19] H. Tang, W. Zhang, X. Li, and S. Wei, "A discrete group teaching optimization algorithm for solving many-objective sand casting whole process production scheduling problem," *Computers & Operations Research*, vol. 164, p. 106563, 2024. doi: 10.1016/j.cor.2024.106563

[20] F. Dugardin, F. Yalaoui, and L. Amodeo, "New multi-objective method to solve reentrant hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 203, no. 1, p. 22–31, 2010. doi: 10.1016/j.ejor.2009.06.031

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi objective optimization: Nsga-ii," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 2000, p. 849–858.

[22] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Tech. Rep. 103, 2001.

[23] F. Dugardin, F. Yalaoui, and L. Amodeo, "Hybrid multi-objective methods to solve reentrant shops," *International Journal of Applied Logistics*, vol. 3, no. 4, p. 15–32, 2012. [Online]. Available: https://ideas.repec.org/a/igg/jal000/v3y2012i4p15-32.html

[24] M. Liu, X. Liu, F. Zheng, and F. Chu, "Bi-objective optimization of a reentrant flow shop scheduling with exact time lag considering energy cost," in *7th International Conference on Industrial Engineering and Systems Management (IESM 2017)*, 2017.

[25] Z. Pan, D. Lei, and L. Wang, "A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling," *IEEE Transactions on Cybernetics*, vol. 52, no. 6, p. 5051–5063, 2022. doi: 10.1109/TCYB.2020.3026571

[26] F. Zhao, S. Di, and L. Wang, "A hyperheuristic with q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, p. 3337–3350, 2023. doi: 10.1109/TCYB.2022.3192112

[27] M. Nedjai, K. Amrouche, and M. Boudhar, "The two-stage chain reentrant hybrid flow-shop problem with deteriorating jobs," in *Proceedings of the 50th Annual Conference of the Operations Research Society of South Africa*, 2021, p. 85.

[28] S. Browne and U. Yechiali, "Scheduling deteriorating jobs on a single processor," *Operations Research*, vol. 38, no. 3, p. 495–498, 1990. doi: 10.1287/opre.38.3.495

[29] B. Alidaee and N. K. Womer, "Scheduling with time-dependent processing times: Review and extensions," *Journal of the Operational Research Society*, vol. 50, p. 711–720, 1999. doi: 10.1057/palgrave.jors.2600740

[30] G. Mosheiov, "Scheduling jobs under simple linear deterioration," *Computers & Operations Research*, vol. 21, no. 6, p. 653–659, 1994. doi: 10.1016/0305-0548(94)90080-9

[31] M. Ji and T. Cheng, "Parallel-machine scheduling of simple linear deteriorating jobs," *Theoretical Computer Science*, vol. 410, no. 38-40, p. 3761–3768, 2009. doi: 10.1016/j.tcs.2009.04.018

[32] L. Wang, X. Huang, P. Ji, and E. Feng, "Unrelated parallel-machine scheduling with deteriorating maintenance activities to minimize the total completion time," *Optimization Letters*, vol. 8, p. 129–134, 2014. doi: 10.1007/s11590-012-0472-x

[33] M. Tigane, M. Dahane, and M. Boudhar, "Multiobjective approach for deteriorating jobs scheduling for a sustainable manufacturing system," *The International Journal of Advanced Manufacturing Technology*, vol. 101, p. 1939–1957, 2019. doi: 10.1007/s00170-018-3043-1

[34] R. Graham, E. Lawler, J. Lenstra, and A. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, p. 287–326, 1979. doi: 10.1016/S0167-5060(08)70356-X

[35] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, p. 182–197, 2002. doi: 10.1109/4235.996017

[36] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: Amosa," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, p. 269–283, 2008. doi: 10.1109/TEVC.2007.900837

[37] C. R. Raquel and P. C. Naval, "An effective use of crowding distance in multiobjective particle swarm optimization," in *GECCO'05: Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, p. 257–264.

[38] C. Hwang and A. Masud, *Multiple objective decision making — methods and applications: A state-of-the-art survey*, ser. Lectures Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 1979. ISBN 978-3-540-09111-0

[39] A. Jaszkiewicz, "Evaluation of multiple objective metaheuristics," in *Metaheuristics for Multiobjective Optimisation*, ser. Lecture Notes in Economics and Mathematical Systems, X. Gandibleux, M. Sevaux, K. Sorensen, and V. T'kindt, Eds. Springer Berlin Heidelberg, 2004, vol. 535, pp. 65–89. ISBN 978-3-642-17144-4