


Article

Hybrid Flow Shop with Unrelated Machines, Setup Time, and Work in Progress Buffers for Bi-Objective Optimization of Tortilla Manufacturing

Victor Hugo Yaurima-Basaldúa ¹, Andrei Tchernykh ^{2,3,*}, Francisco Villalobos-Rodríguez ¹ and Ricardo Salomon-Torres ¹ 

¹ Software Engineering, Sonora State University, San Luis Rio Colorado, Sonora 83455, Mexico; victor.yaurima@ues.mx (V.H.Y.-B.); fco.vr1@gmail.com (F.V.-R.); ricardo.salomon@uabc.edu.mx (R.S.-T.)

² Computer Science Department, CICESE Research Center, Ensenada 22860, Mexico

³ School of Electrical Engineering and Computer Science, South Ural State University, Chelyabinsk 454080, Russia

* Correspondence: tchernykh@cicese.mx; Tel.: +521-646-178-6994

Received: 27 February 2018; Accepted: 1 May 2018; Published: 9 May 2018



Abstract: We address a scheduling problem in an actual environment of the tortilla industry. Since the problem is NP hard, we focus on suboptimal scheduling solutions. We concentrate on a complex multistage, multiproduct, multimachine, and batch production environment considering completion time and energy consumption optimization criteria. The production of wheat-based and corn-based tortillas of different styles is considered. The proposed bi-objective algorithm is based on the known Nondominated Sorting Genetic Algorithm II (NSGA-II). To tune it up, we apply statistical analysis of multifactorial variance. A branch and bound algorithm is used to assert obtained performance. We show that the proposed algorithms can be efficiently used in a real production environment. The mono-objective and bi-objective analyses provide a good compromise between saving energy and efficiency. To demonstrate the practical relevance of the results, we examine our solution on real data. We find that it can save 48% of production time and 47% of electricity consumption over the actual production.

Keywords: multiobjective genetic algorithm; hybrid flow shop; setup time; energy optimization; production environment

1. Introduction

Tortillas are a very popular food as a favorite snack and meal option in various cultures and countries. There are two types of tortillas: wheat-based and corn-based. Their overall consumption is growing all over the world. Originally, tortillas were made by hand: grinding corn into flour, mixing the dough, and pressing to flatten it. In fact, many small firms today still use processes that are more labor-intensive, requiring people to perform a majority of the tasks such as packaging, loading, baking, and distributing. This trend is expected to change over the next years. Due to its considerable practical significance, optimization of tortilla production is important. To improve the production timing parameters (setup, changeover, waiting), operational cost (energy consumption, repairing, service), throughput, etc., careful analysis of the process and of advance scheduling approaches is needed.

This industry is a typical case of a hybrid flow shop, which is a complex combinatorial optimization problem that arises in many manufacturing systems. In the classical flow shop, a set of jobs has to pass through various stages of production. Each stage can have several machines. There is also the flexibility of incorporating different-capacity machines, turning this design into hybrid flow shop scheduling (HFS).

The production decisions schedule work to machines in each stage to determine the processing order according to one or more criteria. Hybrid flow shop scheduling problems are commonly encountered in manufacturing environments. A large number of heuristics for different HFS configurations considering realistic problems have been proposed [1–6].

In spite of significant research efforts, some aspects of the problem have not yet been sufficiently explored. For example, previous works deal mostly with improving the efficiency of production, where objective functions are usually used to minimize the total production time (makespan), the total tardiness, etc. [7]. Although the production efficiency is essential, it is not the only factor to consider in manufacturing operations. One of the main motivations of this work is to reduce energy consumption. In recent years, it has been recognized that high energy consumption can increase operational cost and cause negative environmental impacts [8].

In this paper, we propose a genetic algorithm that contributes to the solution of a bi-objective HFS problem of a real tortilla production environment, with unrelated machines, setup time, and work in progress buffers taking into account the total completion time and energy consumption of the machines.

The rest of the paper is organized as follows. After presenting the description and characteristics of the production line and jobs in Section 2, we discuss related work in Section 3. We describe the model of the problem in Section 4. We present the details of the proposed solution in Section 5 and the algorithm calibration in Section 6. We discuss computational experiments and results in Section 7. Finally, we conclude with a summary and an outlook in Section 8.

2. Production Process

Figure 1 shows the eight main stages of the production process: (1) Preparation, (2) Mixing, (3) Division and Rounding, (4) Repose, (5) Press, (6) Bake, (7) Cooling, (8) Stacking and Packing.

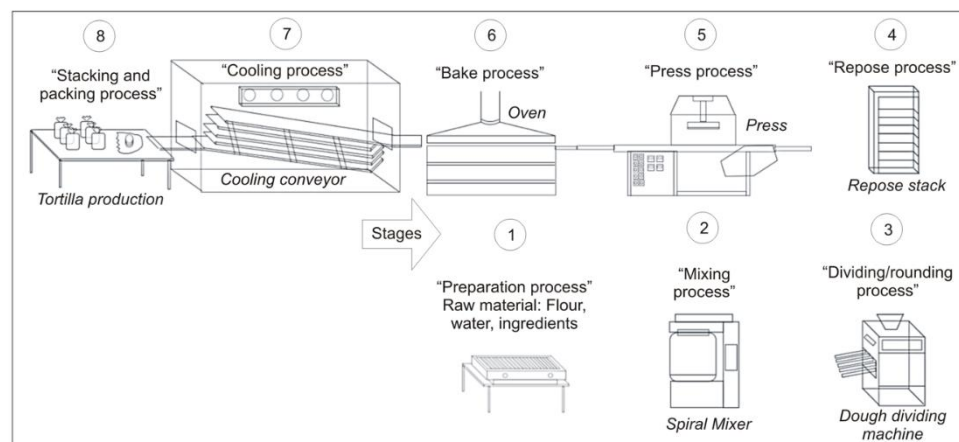


Figure 1. Tortilla production line.

In this paper, we consider a hybrid flow shop with six stages (from 2 to 7), with different numbers of machines in the stages 2, 4, and 6. In the production of wheat flour tortillas, the raw ingredients prepared in Stage 1 are transferred to a large mixer (Stage 2), where they are combined to create the dough.

Corn tortilla production starts with a process called nixtamalization, in which the corn is soaked in an alkali solution of lime (calcium hydroxide) and hot water. The dough is divided into fragments with round shapes (Stage 3). These fragments repose in a controlled environment to obtain a certain consistency as to the temperature and humidity (Stage 4). Next, they are pressed to take the shape of a tortilla (Stage 5). Rows of tortillas are heated in stoves (Stage 6). The next steps are to cool (Stage 7), stack, and pack them (Stage 8).

The process is distributed among flow lines with several machines per stage (Figure 2). It is operated in a semi-automated way controlled by operators.

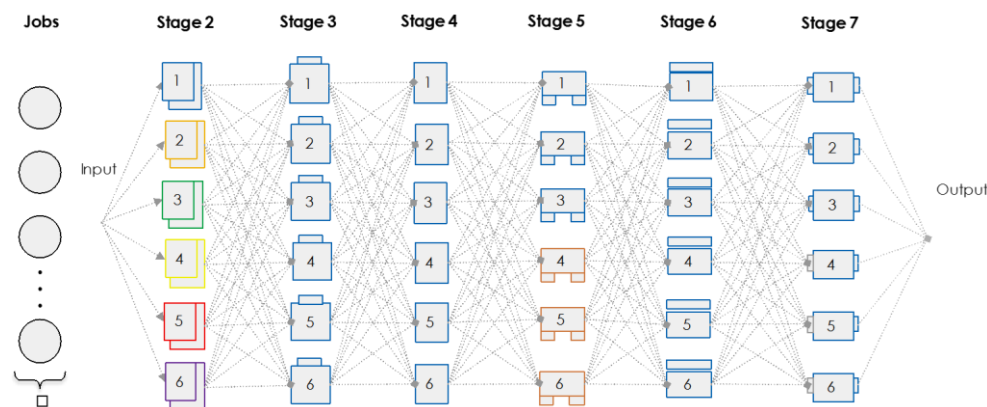


Figure 2. Six-stage hybrid flow shop.

Tables 1 and 2 present the capacities of the machines in each stage. These are represented by the amount of work (tortillas in kilograms) that can be processed on each machine.

Table 1. The processing capacity of machines in Stage 2 (kg per minute).

Stage	2					
Machines	1	2	3	4	5	6
Jobs 1–5	80	80	100	130	160	200

Table 2. The processing capacity of machines in Stages 3–7 (kg per minute).

Stage	3	4	5	6	7	
Machines	1-6	1-6	1-3	4-6	1-6	1-6
Jobs	1	1.3	200	7.2	9.6	9.6
	2	1.3	200	6.4	8.8	8.8
	3	0.44	200	2.4	4	4
	4	0.31	200	3.2	4.8	4.8
	5	1.3	200	4	8.8	8.8

Table 1 demonstrates that Stage 2 has six machines with processing capabilities of 80, 80, 100, 130, 160, and 200 kg per minute that can process jobs from 1 to 5. Machines 1 and 2 have the same capacities.

Table 2 shows that Stages 3, 4, 6, and 7 have identical machines, and Stages 2 and 5 have unrelated machines. Table 3 shows the power of the machines in each stage in kW per minute. The consumption is independent of the type of job to be processed. Machines in Stage 4 do not consume energy. The equipment is cabinets, where dough balls remain for a given period.

Table 3. Energy consumption of machines (kW per minute).

Stage	2	3	4	5	6	7
Machine	1	0.09	0.03	0	0.62	0.04
	2	0.09	0.03	0	0.62	0.04
	3	0.11	0.03	0	0.62	0.04
	4	0.14	0.03	0	0.9	0.04
	5	0.23	0.03	0	0.9	0.04
	6	0.23	0.03	0	0.9	0.04

Table 4 shows characteristics of the jobs. We consider a workload with five jobs for production of the five types of tortillas most demanded by industry: Regular, Integral, Class 18 pcs, Taco 50, and MegaBurro. Each job is characterized by its type, group, and batches required for production. It has several assigned batches. This quantity must be within a certain range that is obtained from the standard deviations of the average monthly production in kilograms of tortillas for a year.

Table 4. Job characteristics.

Job j	Name	Type	Group	Batch Limits
1	Regular	1	1	356–446
2	Integral	2	2	34–61
3	Class 18 pcs	3	1	7–10
4	Taco 50	4	1	13–22
5	MegaBurro	5	1	12–20

The batch indicates the number of 10 kg tasks that have to be processed in the job. A job with one batch indicates processing 10 kg, and with five batches indicates processing 50 kg.

The group indicates the compatibility of the job tasks. Job tasks belonging to the same group may be processed simultaneously. For instance, they can be processed on all unrelated machines of Stage 2, if their sizes are less than or equal to the capacity in kilograms of the machines (Table 1). The job tasks of the same type of dough can be combined, forming compatibility groups. In this case, the jobs of Types 1, 3, 4, and 5 belong to Group 1, and jobs of Type 2 belong to Group 2.

Table 5 shows processing times in minutes for every 10 kg of tortilla for a job type. The 10 kg represents a task. We see that in Stages 2 and 4, all machines process jobs for the same durations of 1 and 45 min, respectively, due to the fact that the time does not depend on the job type.

Table 5. Processing time $P_{ik,j}$ of job j at machine k in stage i for each 10 kg.

Stage	2	3	4	5	6	7		
Machines	1-6	1-6	1-6	1-3	4-6	1-6	1-6	
Jobs	1	1	1.25	45	1.43	1.07	12.15	17.25
	2	1	1.25	45	1.48	1.11	12.15	17.25
	3	1	4.38	45	3.69	2.77	21.38	26.48
	4	1	3.13	45	3.00	2.25	24.50	29.60
	5	1	0.42	45	2.28	1.14	18.24	23.34

Table 6 shows the setup time of machines at each stage. At stages from 3 to 7, the setup time is independent of the task sequence, job type, and weight.

Table 6. Sequence setup time for all machines in stage i .

Stage	2	3	4	5	6	7
Setup time	st_2	0.7	0.7	0.5	0	0

At Stage 2, the setup time is calculated as follows:

$$st_2 = \frac{\left(35 + \left(\frac{t_w}{10} \times 5\right)\right)}{60} \quad (1)$$

where t_w is the total weight in kilograms of the tasks to be processed by machines in the same time period. These tasks have to belong to jobs of the same group.

3. Related Works

3.1. Energy-Aware Flow Shop Scheduling

Investment in new equipment and hardware can certainly contribute to energy savings [9]. The use of “soft” techniques to achieve the same objective is also an effective option [10].

The advance schedule could play an important role in reducing the energy consumption of the manufacturing processes. Meta-heuristics, e.g., genetic algorithms (GA) [11], particle swarm optimization [12], and simulated annealing [13] are greatly popularity for use in the design of production systems.

However, studies of flow shop scheduling problems for energy saving appear to be limited [14,15]. In fact, most production systems that allow changing of the speed of the machines belong to the mechanical engineering industry and usually take the configuration of a work shop instead of a flow shop [16].

One of the first attempts to reduce energy consumption through production scheduling can be found in the work of Mouzon et al. [17]. The authors collected operational statistics for four machines in a flow shop and found that nonbottleneck machines consume a considerable amount of energy when left idle. As a result, they propose a framework for scheduling power on and off events to control machines for achieving a reduction of total energy consumption. Dai et al. [18] applied this on/off strategy in a flexible flow shop, obtaining satisfactory solutions to minimize the total energy consumption and makespan.

Zhang and Chiong [16] proposed a multiobjective genetic algorithm incorporated into two strategies for local improvements to specific problems, to minimize power consumption, based on machine speed scaling.

Mansouri et al. [19] analyzed the balance between minimizing the makespan, a measure of the level of service and energy consumption, in a permutation flow shop with a sequence of two machines. The authors developed a linear model of mixed-integer multiobjective optimization to find the Pareto frontier composed of energy consumption and total makespan.

Hecker et al. [20] used evolutionary algorithms for scheduling in a non-wait hybrid flow shop to optimize the allocation of tasks in production lines of bread, using particle swarm optimization and ant colony optimization. Hecker et al. [21] used a modified genetic algorithm, ant colony optimization, and random search procedure to study the makespan and total time of idle machines in a hybrid permutation flow model.

Liu and Huang [14] studied a scheduling problem with batch processing machines in a hybrid flow shop with energy-related criteria and total weighted tardiness. The authors applied the Nondominated Sorting Genetic Algorithm II (NSGA-II).

Additionally, in time completion problems, Yaurima et al. [3] proposed a heuristic and meta-heuristic method to solve hybrid flow shop (HFS) problems with unrelated machines, sequence-dependent setup time (SDST), availability constraints, and limited buffers.

3.2. Multiobjective Optimization

Different optimization criteria are used to improve models of a hybrid flow shop. An overview of these criteria can be found in [4]. Genetic algorithms have received considerable attention as an approach to multiobjective optimization [6,22,23].

Deb et al. [24] proposed a computationally efficient multiobjective algorithm called NSGA-II (Nondominated Sorting Genetic Algorithm II), which can find Pareto-optimal solutions. The computational complexity of the algorithm is $O(MN^2)$, where M is the number of targets and N is the size of the data set.

In this article, we consider it to solve our problem with two criteria. An experimental analysis of two crossover operators, three mutation operators, three crossover and mutation probabilities,

and three populations is presented. For each case of machines per stage, the goal is to find the most desirable solutions that provide the best results considering both objectives.

3.3. Desirability Functions

Getting solutions on the Pareto front does not resolve the multiobjective problem. There are several methodologies to incorporate preferences in the search process [25].

These methodologies are responsible for giving guidance or recommendations concerning selecting a final justifiable solution from the Pareto front [26]. One of these methodologies is Desirability Functions (DFs) that are responsible for mapping the value of each objective to desirability, that is, to values in a unitless scale in the domain $[0, 1]$.

Let us consider that the objective f_i is $Z_i \subseteq \mathbb{R}$; then a DF is defined as any function $d_i : Z_i \rightarrow [0, 1]$ that specifies the desirability of different regions of the domain Z_i for objective f_i [25].

The Desirability Index (DI) is the combination of the individual values of the DFs in one preferential value in the range $[0, 1]$. The higher the values of the DFs, the greater the DI. The individual in the final front with the highest DI will be the best candidate to be selected by the decision maker [27]. The algorithm calibration of desirability functions is applied to the final solutions of the Pareto front to get the best combination of parameters.

4. Model

Many published works address heuristics for flow shops that are considered production environments [3,23,28–30]. We consider the following problem: A set $J = \{1, 2, 3, 4, 5\}$ of jobs available at time 0 must be processed on a set of 6 consecutive stages $S = \{2, 3, 4, 5, 6, 7\}$ of the production line, subject to minimization of total processing time C_{max} and energy consumption of the machines E_{op} .

Each stage $i \in S$ consists of a set $M_i = \{1, \dots, m_i\}$ of parallel machines, where $|M_i| \geq 1$. The sets M_2 and M_5 have unrelated machines and the sets M_3, M_4, M_6 , and M_7 have identical machines. We consider three different cases of $m_i = \{2, 4, 6\}$ machines in each stage.

Each job $j \in J$ belongs to a particular group G_k . Jobs $1, 3, 4, 5 \in J$ belong to G_1 , job $2 \in J$ belongs to G_2 . Each job is divided into a set T_j of t_j tasks, $T_j = \{1, 2, \dots, t_j\}$. Jobs and tasks in the same group can be processed simultaneously only in stage $2 \in S$ according to the capacity of the machine. In stages $3, 4, 5, 6, 7 \in S$, the tasks must be processed by exactly one machine in each stage (Figure 2).

Each task $t \in T_j$ consists of a set of batches of 10 kg each. The tasks can have different numbers of batches. The set of all the tasks together of all the jobs is given by $T = \{1, 2, \dots, Q\}$.

Each machine must process an amount of “ a ” batches at the time of assigning a task.

Let $p_{i_l,q}$ be the processing time of the task $q \in T_j$ on the machine $l \in M_i$ at stage i . Let $S_{i_l,qp}$ be the setup (adjustment) time of the machine l from stage i to process the task $p \in T_j$ after processing task q . Each machine has a process buffer for temporarily storing the waiting tasks called “work in progress”. The setup times in stage $2 \in S$ are considered to be sequence dependent; in stages $3, 4, 5, 6, 7 \in S$, they are sequence independent.

Using the well-known three fields notation $\alpha \mid \beta \mid \gamma$ for scheduling problems introduced in [31], the problem is denoted as follows:

$$FH6, (RM^{(2)}, (PM^{(i)})_{i=3^4}, RM^{(5)}, (PM^{(i)})_{i=6^7}) \mid S_{sd2}, S_{si3^7} \mid \{C_{max}, E_{op}\}. \quad (2)$$

Encoding

Each solution (chromosome) is encoded as a permutation of integers. Each integer represents a batch (10 kg) of a given job. The enumeration of each batch is given in a range specified by the order of the jobs and the number of batches.

Batches are listed according to the job to which they belong. Table 7 shows an example the five jobs. If Job 1 has nine batches, they are listed from 1 to 9. If Job 2 has 12 batches, they are listed from 10 to 21. Reaching Job 5, we have accumulated 62 batches.

Table 7. Example of enumeration of batches for jobs.

Job	1	2	3	4	5
Batches	9	12	16	15	10
Batch index	1–9	10–21	22–37	38–52	53–62

Figure 3 shows an example of the production representation. Groups are composed according to the number of machines in the stage (m_i). The batches of each job are randomly distributed between the groups. To obtain the number of batches per group (a), we divide the total amount of batches by the number of machines per stage.

Machine 1					Machine 2						Machine m_i				
56	23	...	19	16	1	55	...	62	27	...	11	24	...	37	17
1	2	...	$a - 1$	a	1	2	...	$a - 1$	a		1	2	...	$a - 1$	a

Figure 3. Example of the representation of the chromosome for m_i machines per stage.

Let us consider six stages with $m_1 = 4$ machines per stage. The chromosome is composed of 4 groups. Let us assume that we have 62 batches for 5 jobs. This is equal to 62 (batches)/4 (machines per stage) = 15.5 (batches per group). The result is rounded to the nearest higher integer, ending with $a = 16$ batches per group. An example of processing tasks on Machine 1 is shown in Figure 4.

	Machine 1																
Original Sequence	56	23	38	35	1	42	61	14	51	11	5	59	48	15	19	16	Batches
Ordered Sequence	1						2				3				Proc. Order		
	1	5	23	35	38	42	51	48	56	61	59	14	11	15	19	16	Batches
	1	3	4			4	5		2			Jobs					

Figure 4. Example of tasks on Machine 1.

In the ordered sequence, “Jobs” indicate jobs to which the batches belong according to their enumeration. These groups indicate the number of total tasks for each job from left to right. For example, in the ordered sequence, Job 4 has two tasks; the first one “4-1” has three batches (38, 42, 51) and the second one “4-2” has one batch (48).

“Proc. Order” indicates the order of processing of the task groups. Assuming that a machine in Stage 1 has to process up to 70 kg (7 batches), Group 1 is composed of sets of tasks 1-1 (1, 5), 3-1 (23, 35), 4-1 (38, 42, 51) with seven batches. When the capacity of the machine is filled, Group 2 is formed for the tasks 4-2 (48) and 5-1 (56, 61, 59). Assuming that the batches of Job 2 are not compatible with any of the remaining four jobs, the batches are processed separately from the other two groups, forming the task 2-1 with five batches (14, 11, 15, 19, 16). When the first group’s tasks have completed their processing in the machine, the second group is processed, and so on.

The tasks processed in the group are assigned to the machines of the following stages individually according to their order of completion in the first stage. When the tasks of a group—for example, Group 1 (1-1, 3-1, 4-1)—are processed together, the completion time is the same for all. Therefore,

the three tasks will be ready at the same time to be assigned individually to machines available in the next stage, until the last (6th) stage.

In the post-second stages, the tasks that are ready to be assigned are in the process buffer. The task that has the longest processing time is assigned first to an available machine. The calculation of the total completion time C_{max} is as follows. The time $C_{i,p}$ for completing task p in stage i is calculated according to the formula

$$C_{i,p} = \min_{1 \leq l \leq m} \{ \max \{ C_{i,p} + S_{i,l,p}; C_{i-1,p} \} + p_{i,l} \}. \quad (3)$$

The maximum completion time of all tasks and jobs is calculated as

$$C_{max} = \max_{p=1}^Q \{ C_{7,p} \}. \quad (4)$$

Q indicates the total number of tasks for all jobs. $C_{m,p}$ is the completion time of task $p \in T_j$ in the last stage $7 \in S$. The total energy consumption E_{op} of the execution of all tasks is calculated as

$$E_{op} = \sum_{q=1}^Q p_{i_l,q} \cdot E_{i_l}. \quad (5)$$

where E_{i_l} indicates the electrical power consumption of machine l in stage i , and $p_{i_l,q}$ refers to the processing time of the task $q \in T$.

5. Bi-Objective Genetic Algorithm

5.1. Nondominated Sorting Genetic Algorithm II (NSGA-II)

The NSGA-II algorithm is used to assign tasks to machines so that C_{max} and E_{op} are minimized. NSGA-II is a multiobjective genetic algorithm characterized by elitism and stacking distance to maintain the diversity of the population to find as many Pareto-optimal solutions as possible [24]. It generates a population of N individuals, where each represents a possible solution. Individuals evolve through genetic operators to find optimal or near-optimal solutions. Three operators are usually applied: tournament selection (using a stacking tournament operator), crossover, and mutation to generate another N individuals or children.

From the mixture of these two populations, a new population of size $2N$ is created. Then, the best individuals are taken according to their fitness value by ordering the population on nondominated fronts. Individuals from the best nondominated fronts are first taken, one by one, until N individuals are selected.

The crowding distance is then compared to preserve diversity in the population. This operator compares two solutions and chooses a tournament winner by selecting the setting that is located on the best Pareto front. If the participating tournament configurations are on the same front, the best crowding distance (highest) to determine the winning setting is used. Later, the algorithm applies the basic genetic operators and promotes the next generation cycle with the configurations that occupy the best fronts, preserving the diversity through the crowding distance. A job is assigned to a machine at a given stage by taking into consideration processing speeds, setup times, machine availability, and energy consumption. Each solution is encoded in a permutation of integers.

5.2. Crossover Operators

Crossover operators allow the obtaining of new solutions (children) by combining individuals (parents) of the population. The crossover operator is applied under a certain probability. In this paper, we consider two operators: the partially mapped crossover and the order crossover.

Partially mapped crossover (PMX) [32,33]: This operator uses two cut points (Figure 5). The part of the first parent between the two cut points is copied to the children. Then, the following parts of the children are filled by a mapping between the two parents, so that their absolute positions are inherited where possible from the second parent.

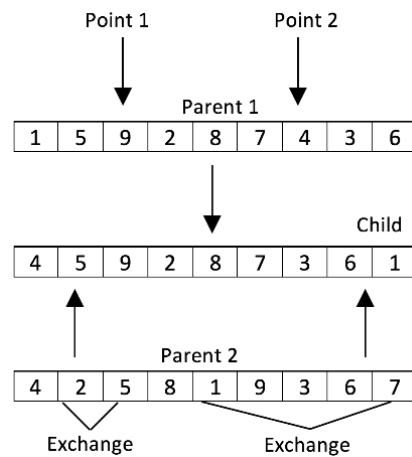


Figure 5. Partially mapped crossover (PMX) operator.

Figure 6 shows an example of the PMX crossover. The cross points in both parents serve to form the child chromosome. The 16 listed lots of the 4 jobs are randomly distributed on each parent chromosome. When crossed, they form the child chromosome with 16 lots.

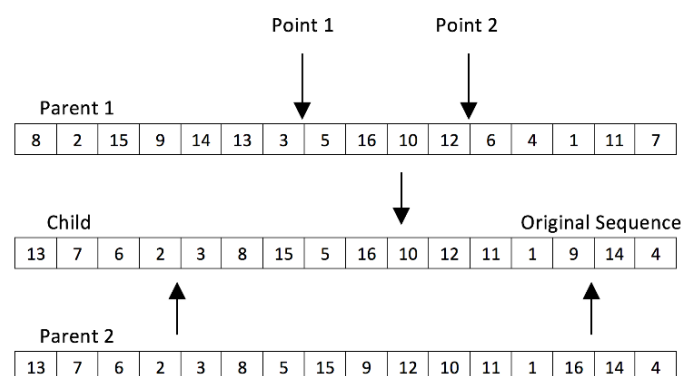


Figure 6. PMX crossover operator example with batch index.

The cross parents are composed of the batches of the jobs listed in Table 8.

Table 8. Enumeration of the batches for jobs.

Job	Batches	Batch Index
1	4	1–4
2	4	5–8
3	4	9–12
4	4	13–16

Figures 7 and 8 show representations of tasks for two and four machines per stage. The original sequence is ordered considering the enumeration of the jobs and their batches. Each machine is assigned batches to process, which form the tasks of each job. Each machine must process a certain set

of tasks; each set is assigned a processing order (Proc. Order). These sets are formed according to the processing compatibility.

	m_1							m_2									
Ordered	1			2				1					2		Proc. Order		
	2	3	13	15	5	6	7	8	1	4	9	10	11	12	14	16	Batch
	1		4		2		1		3			4		Jobs			

Figure 7. Chromosome representation example with two machines per stage.

	m_1				m_2				m_3				m_4				
	1				2				1				1				Proc. Order
Ordered	2	3	6	7	3	15	5	8	10	11	12	16	1	4	9	14	Batch
	1		4		2		1		4		2		3		4		Jobs

Figure 8. Chromosome representation example with four machines per stage.

The tasks of Jobs 1, 3, and 4 can be processed in the same set at the same time. The tasks of Job 2 are put into a separate set because they are not compatible with each other. For example, in Figure 7, the first two tasks (1-1, 4-1) are processed by Machine 1, then the second set with one task (2-1). In Machine 2, two tasks of the first set (1-2 and 3-1) are processed, assuming that the machine has only the capacity to process up to 6 batches (60 kg). The second set will then be followed by one task (4-1).

Figure 8 shows four machines per stage. Four batches are assigned to each machine. Machines 1 and 2 process two sets of tasks. Machines 3 and 4 process only one set. Upon completion of the tasks processed in sets in Step 1, in the later stages, they will be processed individually.

Order crossover (OX) [32,34–37]: Two cut points are selected randomly; the part of the first parent located between these two cut points is copied to the child. The child's other spaces are left blank. The values copied to the child are deleted in the second parent. The following positions in the child are filled starting in the blank spaces and considering the order found in the second parent (Figure 9).

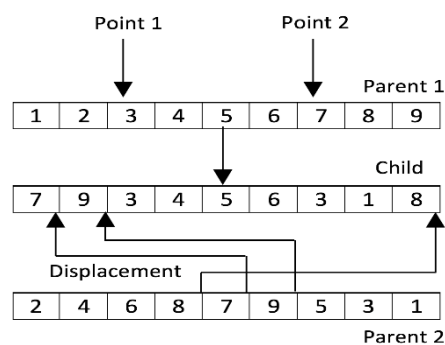


Figure 9. Order crossover (OX) operator.

5.3. Mutation Operators

Mutation operators produce small changes in individuals according to a probability. This operator helps to prevent falling into local optima and to extend the search space of the algorithm. Three mutation operators are considered: Displacement, Exchange, and Insertion.

Displacement is a generalization of the insertion mutation, in which instead of moving a single value, several values are changed (Figure 10). The exchange operator selects two random points and these position values are exchanged (Figure 11). In insertion, a value is selected randomly and will be inserted at an arbitrary position (Figure 12).

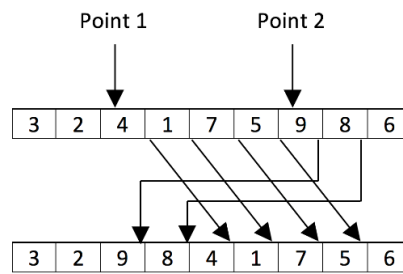


Figure 10. Displacement mutation operator.

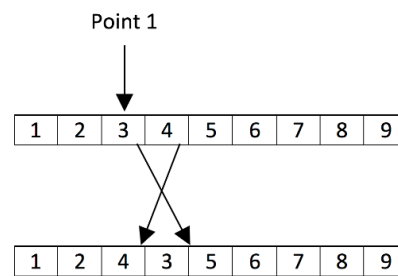


Figure 11. Exchange mutation operator.

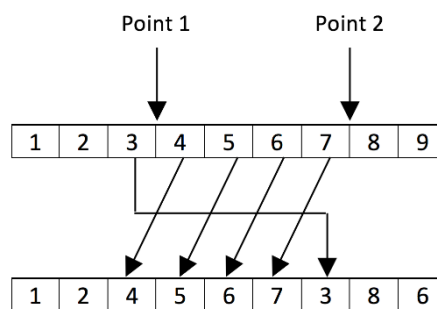


Figure 12. Insertion mutation operator.

6. Tuning Up the Parameters

6.1. Parameter Calibration

Calibration is a procedure for choosing the algorithm parameters that provide the best result in the response variables. A computational experiment for calibration includes the following steps: (a) each instance or workload is run with all possible combinations of parameters; (b) the best solution is obtained; (c) the relative difference of each algorithm on the best solution is calculated; (d) multifactorial Analysis of Variance (ANOVA) with a 95% confidence level is applied to find the parameter that most influences the solution; (e) the set of the best values is selected for each parameter.

Table 9 shows the parameters used for calibration. A total of $3 \times 2 \times 3 \times 3 \times 3 = 162$ different algorithms or combinations are considered. Thirty runs for each combination are performed, with $162 \times 30 = 4860$ experiments in total. For each of the 30 runs, a different workload will be taken. For each of the five jobs, 30 batches are generated randomly from a uniform distribution according to their limits (Table 4).

Each batch is assigned to a workload, obtaining 30 loads from 5 jobs. The variation of processing capabilities (in minutes) of the machines in Stages 2 and 5 is generated according to Tables 1 and 2. Stage 2 has 6 machines with different processing capabilities in kilograms. Stage 5 has 6 machines.

According to the selection of a number of machines per stage, the set of machines is chosen. The machines are numbered from 1 to 6. By considering different numbers of machines per stage (2, 4, or 6), we conduct $4860 \times 3 = 14,580$ experiments. In each of the 30 runs of each combination, the best individual applying desirability function is obtained. Subsequent values of each objective (C_{max} and E_{op}) are used to calculate an average of 30, obtaining a single value for each objective.

Table 9. Parameters used for calibration.

Factors	Levels
Population	20, 30, 50
Crossover operators	OX, PMX
Crossover probability	0.5, 0.7, 0.9
Mutation operators	Displacement, Exchange, Insertion
Mutation probability	0.05, 0.1, 0.2
Selection	Binary tournament
Stop criterion	50 iterations

The performance of each algorithm is calculated as the relative increment of the best solution (RIBS). The RIBS is calculated with the following formula:

$$RIBS = \frac{Heu_{sol} - Best_{sol}}{Best_{sol}} \times 100 \quad (6)$$

where Heu_{sol} is the value of the objective function obtained by the algorithm, and $Best_{sol}$ is the best value obtained during the execution of all possible combinations of parameters.

All experiments were performed on a PC with Intel Core i3 CPU and 4 GB RAM. The programming language used to encode the algorithm is R. It provides advantages, including the scalability and libraries [38]. The calibration of the algorithm lasted approximately 15 days and 9 h.

6.2. Residual Analysis

To verify that the data are normalized, the supposition of the suitability of the model using the normality, homoscedasticity, and independence of residues is verified. The residuals are calculated according to the following formula [39]:

$$e_i = y_i - \bar{y}_i, \quad i = 1, 2, 3, \dots, n \quad (7)$$

where y_i is the RIBS for the run i and \bar{y}_i is the average of the experiment. Figure 3 shows the graph of the normal probability of residuals for 6 machines per stage. As can be seen (Figure 13), the graph complies with the assumption of normality. The same results are obtained for 2 and 4 machines per stage.

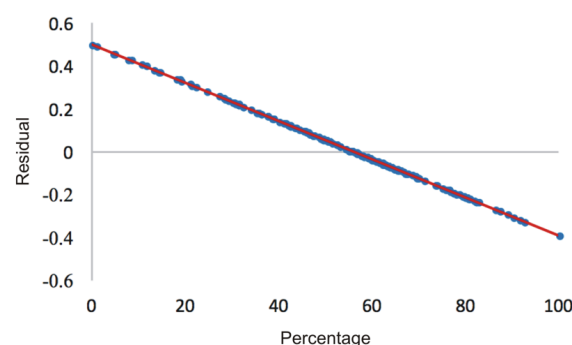


Figure 13. Graph of the normal probability of waste for six machines per stage.

6.3. Variance Analysis

Variance analysis is applied to evaluate the statistical difference between the experimental results and to observe the effect of the parameters on the quality of the results. It is used to determine factors that have a significant effect and to discover the most important factors. The parameters of the problem are considered as factors and their values as levels. We assume that there is no interaction between factors.

The *F*-Ratio is the ratio between Factor Mean Square and Mean Square residue. A high *F*-Ratio means that this factor significantly affects the response. The *p*-value shows the statistical significance of the factors: *p*-values that are less than 0.05 have a statistically significant effect on the response variable (RIBS) with a 95% confidence level. According to the *F*-Ratio and *p*-value, the most important factors in the case of two machines per stage are the mutation operator and the crossover operator variable. DF shows the number of degrees of freedom (Table 10).

Table 10. Relative increment of the best solution (RIBS) analysis of variance for two machines per stage.

Source	Sum of Squares	DF	Mean Square	<i>F</i> -Ratio	<i>p</i> -Value
A: Crossover	1747	1	1747	9.755	0.00213
B: Mutation	11,967	1	11,967	66.833	9.58×10^{-14}
C: Crossover probability	1519	1	1519	8.481	0.00411
D: Mutation probability	654	1	654	3.652	0.05782
E: Population	1732	1	1732	9.673	0.00222
Residuals	27,934	156	179		

In the case of four machines per stage, the most important factors are the mutation operator and crossover probability (Table 11). For six machines per stage, the most important factors are the mutation operator and mutation probability (Table 12).

Table 11. RIBS analysis of variance for four machines per stage.

Source	Sum of Squares	DF	Mean Square	<i>F</i> -Ratio	<i>p</i> -Value
A: Crossover	760	1	760	4.602	0.03349
B: Mutation	13,471	1	13,471	81.565	6.09×10^{-16}
C: Crossover probability	1540	1	1540	31.223	1.00×10^{-7}
D: Mutation probability	5157	1	5157	9.326	0.00266
E: Population	3395	1	3395	20.557	1.15×10^{-5}
Residuals	25,765	156	165		

Table 12. RIBS analysis of variance for six machines per stage.

Source	Sum of Squares	DF	Mean Square	<i>F</i> -Ratio	<i>p</i> -Value
A: Crossover	2587	1	2587	18.39	3.14×10^{-5}
B: Mutation	17,785	1	17,785	126.47	2×10^{-16}
C: Crossover probability	2886	1	2886	20.52	1.16×10^{-5}
D: Mutation probability	4840	1	4840	34.42	2.58×10^{-8}
E: Population	2881	1	2881	20.49	1.18×10^{-5}
Residuals	21,937	156	165		

Figures 14–17 show means and 95% Least Significant Difference (LSD) confidence intervals. Figure 14 shows the results obtained for mutation operators for two, four, and six machines per stage. We can see that operator Displacement is the best mutation among the three tested operators. Figure 15 shows the results for the crossover operators, where we observe that the PMX operator is the best. The results for four and six stages are similar.

Figure 16 shows plots for the crossover probability. It can be seen that the best crossover probability occurring is 0.9. Figure 17 presents plots for the mutation probability, where the probability of 0.2 is statistically more significant.

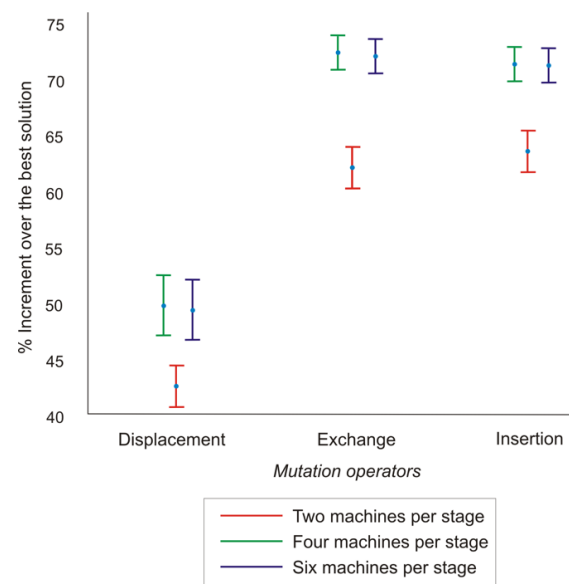


Figure 14. Means and 95% LSD confidence intervals of mutation operators.

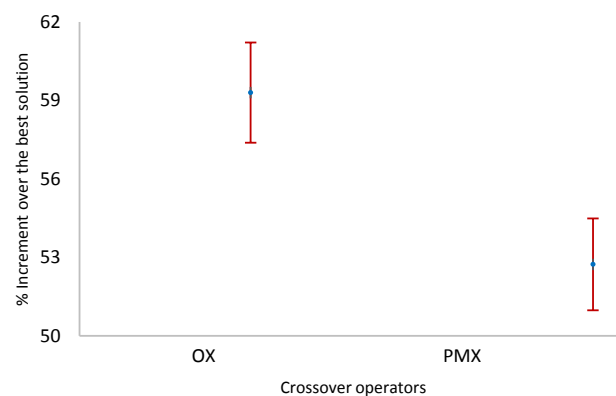


Figure 15. Means and 95% LSD confidence intervals of crossover operators—two machines per stage.

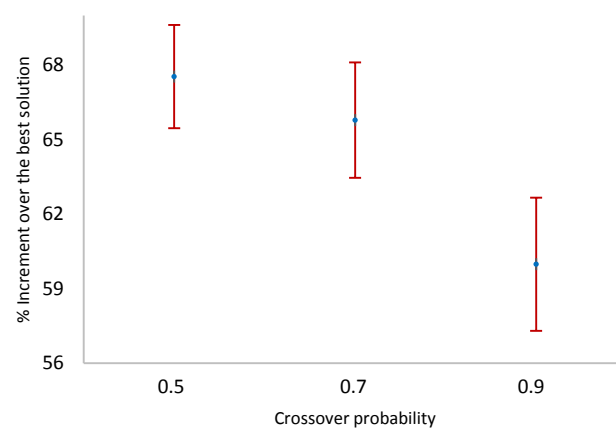


Figure 16. Means and 95% LSD confidence intervals of crossover probability—four machines per stage.

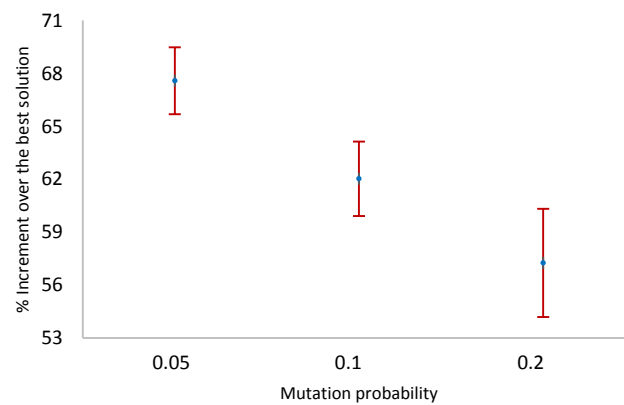


Figure 17. Means and 95% LSD confidence intervals of mutation probability—six machines per stage.

6.4. Pareto Front Calibration Analysis

Figures 18–20 show the solution space obtained by calibration experiments. The horizontal axis represents the energy E_{op} consumed by the machines. The vertical axis represents the time completion C_{max} of jobs. To determine the best individuals of all experiments, the Pareto front is calculated for each case. Each point represents a combination of parameters from 162 experiments explained in Section 6.1. Each front consists of numbered points from lowest to highest according to their DI. The closer to 1, the better the place in the enumeration is (see Section 3.3).

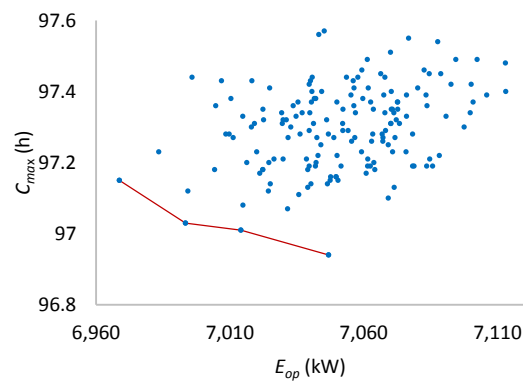


Figure 18. Pareto front for two machines per stage.

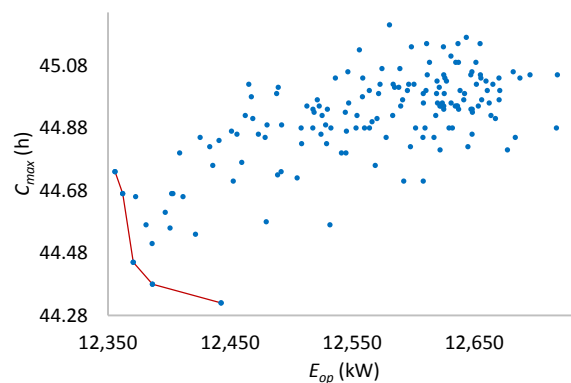


Figure 19. Pareto front for four machines per stage.

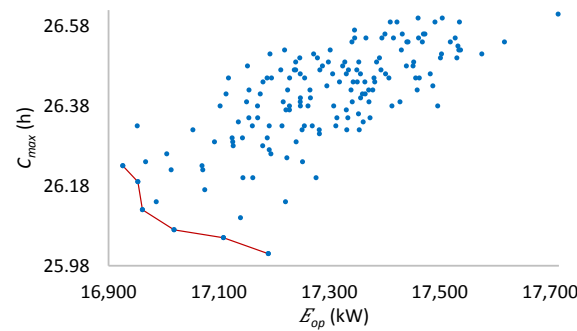


Figure 20. Pareto front for six machines per stage.

Tables 13–15 show the ordering of the front points regarding a combination of E_{op} , C_{max} , and Desirability Index (DI) for each Pareto front. We see, for example, that the parameter combination 156 corresponds to a higher DI, followed by 129, etc.

Table 13. Optimum parameter combination selection for bi-objective genetic algorithm (GA) for two machines per stage.

No.	Comb.	E_{op}	C_{max}	DI
1	156	6993.41	97.03	0.843
2	129	6968.70	97.15	0.816
3	128	7014.08	97.01	0.782
4	120	7046.87	96.94	0.677

Table 14. Optimum parameter combination selection for bi-objective GA for four machines per stage.

N	Comb.	E_{op}	C_{max}	DI
1	156	12,381.55	44.38	0.926
2	129	12,370.82	44.45	0.906
3	147	12,442.75	44.32	0.878
4	128	12,362.17	44.67	0.775
5	120	12,355.79	44.74	0.730

Table 15. Optimum parameter combination selection for bi-objective GA for six machines per stage.

N	Comb.	E_{op}	C_{max}	DI
1	156	17,018.73	26.07	0.891
2	129	16,961.46	26.12	0.883
3	120	17,108.33	26.05	0.847
4	147	16,953.30	26.19	0.823
5	138	17,189.52	26.01	0.816
6	75	16,925.85	26.23	0.796

Table 16 shows the parameters of these combinations. The crossover PMX has higher values of DI. The mutation operator Displacement is used in each of the top three combinations. The crossover probability is maintained at 0.9 in the two most significant combinations. The mutation probability remains at the highest with 0.2 in all three, as does the population with 50 individuals.

The most influential parameters correspond to the combination 156, which has greater DI in three cases (see Tables 13–15).

Our analysis shows that crossover PMX and mutation operator Displacement are the best among those tested. The best probability of crossover is 0.9 and that of mutation is 0.2. The population of 50 individuals is statistically more significant.

Table 16. Optimum parameter combination for bi-objective GA.

No.	Comb.	Cr	Mu	Pcr	Pmu	Pob	
1	156	PMX	Displacement	0.9	0.2	50	Selected
2	129	OX	Displacement	0.9	0.2	50	
3	120	OX	Displacement	0.7	0.2	50	

7. Experimental Analysis

7.1. Experimental Setup

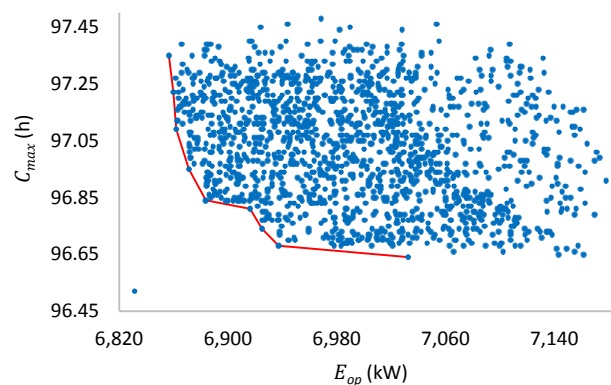
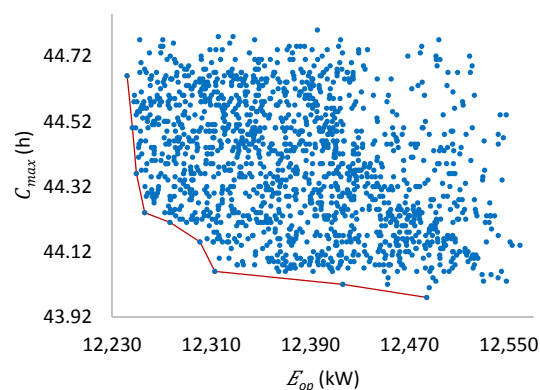
In multiobjective optimization, there is usually no single best solution, but a set of solutions that are equally good. Our objective is to obtain a good approximation of the Pareto front regarding two objective functions [40].

Our bi-objective genetic algorithm is tuned up using the following parameters obtained during the calibration step: crossover, PMX; mutation, Displacement; crossover probability, 0.9; mutation probability, 0.2; population size, 50.

We consider five jobs in each of the 30 workloads, obtaining the number of batches for each job based on a uniform random distribution (Table 4).

7.2. Bi-Objective Analysis

Figures 21–23 show the solution sets and the Pareto fronts. In each front, 1500 solutions are included, being 50 (individuals) \times 30 (workloads).

**Figure 21.** Pareto front for two machines per stage.**Figure 22.** Pareto front for four machines per stage.

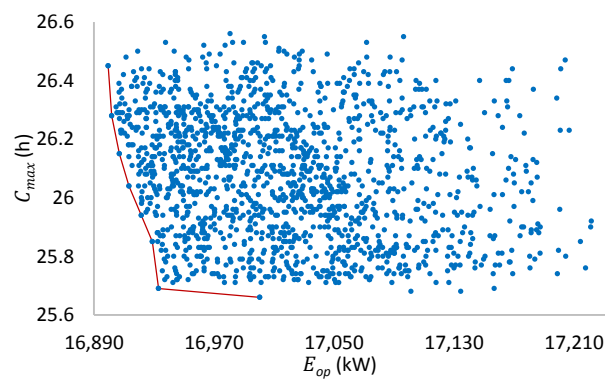


Figure 23. Pareto front for six machines per stage.

Figure 21 shows the solution space (two objectives) for two machines per stage. This two-dimensional solution space represents a feasible set of solutions that satisfy the problem constraints. The Pareto front covers a wide range of E_{op} from 6856 to 7178 kW, while C_{max} is in the range of 96.64 to 97.48 h.

Figure 22 shows the solution space for four machines per stage. The Pareto front covers E_{op} from 12,242 to 12,559 kW, while C_{max} is in the range of 43.98 to 44.8 h. Finally, Figure 23 shows the solution space for 6 machines per stage. E_{op} ranges from 16,899 to 17,221 kW, while C_{max} is in the range of 25.66 to 26.66 h.

Although the Pareto fronts are of good quality, it is observed that many of the generated solutions are quite far from it. Therefore, a single execution of the algorithm can produce significantly different results. The selection of the solutions included in the Pareto front depends on the preference of the decision maker.

7.3. Branch and Bound Analysis

A Branch and Bound algorithm (B&B) is an exact method for finding an optimal solution to an NP-hard problem. It is an enumerative technique that can be applied to a wide class of combinatorial optimization problems. The solution searching process is represented by a branching tree. Each node in the tree corresponds to a subproblem, which is defined by a subsequence of tasks that are placed at the beginning of the complete sequence.

This subsequence is called partial sequence (PS). The set of tasks not included in PS is called NPS. When a node is branched, one or more nodes are generated by adding a task ahead of the partial sequence associated with the node being branched. To avoid full enumeration of all task permutations, the lower bound of the value of the objective functions is calculated in each step for each partial schedule. In the case of ties, the algorithm selects a node with the lowest lower bound.

The B&B algorithm was run on a PC with an Intel Core i3 CPU and 4 GB of RAM. The programming language used to encode the algorithm was R. The algorithm lasted approximately 12 days, 3 h.

A comparison of the B&B algorithm concerning the Pareto front is shown in Figure 21 as a lowest left point. The B&B result is considered as the best result that could be obtained, known as the global optimum.

Table 17 shows the relative degradation of our bi-objective algorithm over the B&B best solutions. Results are obtained by averaging 30 experiments. We see that the results for each objective are not more than 3.44% worst for C_{max} and 2.94% for E_{op} . Table 18 shows the computation times for obtaining the Pareto front solutions.

According to the results comparison, we conclude that the proposed bi-objective algorithm can produce satisfactory results close to a global optimum in short computational time.

Table 17. RIBS of the solution of the Branch and Bound (B&B) algorithm concerning the Pareto front solutions.

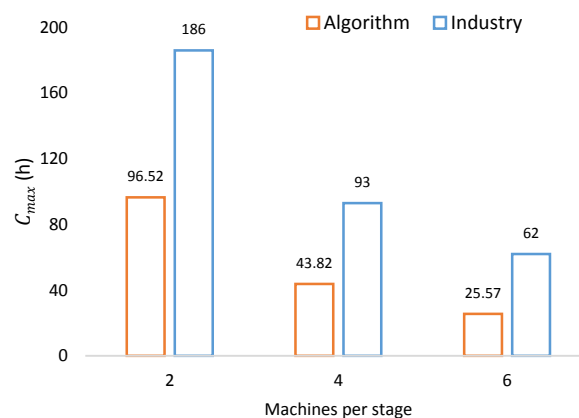
30×2		30×4		30×6	
E_{op}	C_{max}	E_{op}	C_{max}	E_{op}	C_{max}
2.94	0.12	1.72	0.46	0.50	1.8
1.55	0.17	0.87	0.55	1.02	0.35
0.77	0.33	0.41	0.96	0.42	3.44
1.37	0.23	0.33	1.55	0.46	2.27
1.24	0.30	0.58	0.89	0.62	0.47
0.58	0.45	0.77	0.75	0.60	1.1
0.45	0.59	0.35	1.23	0.55	1.45
0.42	0.73	0.29	1.92	0.43	2.78
0.37	0.86	2.27	0.37		

Table 18. Computational time for obtaining Pareto front solutions.

Time (min)		Time (min)		Time (min)	
2 machines per stage	2.85	4 machines per stage	3.77	6 machines per stage	7.55
	3.79		5.81		8.6
	1.74		6.78		6.7
	2.76		5.67		6.6
	3.73		3.85		7.62
	3.86		4.82		7.66
	2.85		4.85		8.61
	1.77		4.94		6.76
	2.85		5.8		

7.4. Comparison with Industry

Figures 24 and 25 show E_{op} and C_{max} of the industry and the B&B algorithm. The term “Algorithm” in these graphs represents the results of the B&B algorithm. The term “Industry” represents the actual data of the production company. The industry results shown in Table 19 are obtained considering the average monthly production load in kilograms of tortillas per year. The results of the B&B algorithm are obtained for a different number of machines per stage. Table 20 shows the same results as in Table 19 in percentages.

**Figure 24.** C_{max} of B&B algorithm and industry.

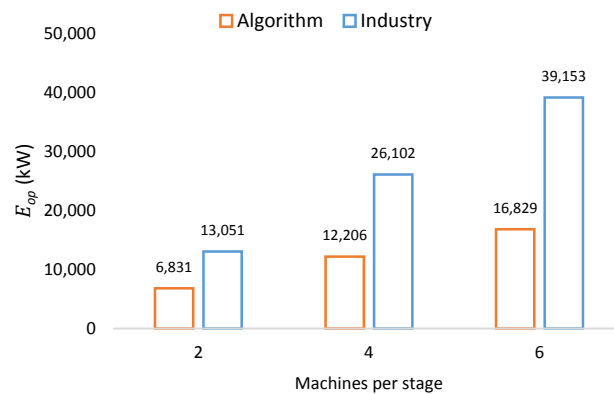


Figure 25. E_{op} of B&B algorithm and industry.

Table 19. Results of the industry and B&B algorithm.

Machines per Stage	Industry		B&B Algorithm	
	E_{op} (kW)	C_{max} (h)	E_{op} (kW)	C_{max} (h)
2	13,050.90	186	6,831.40	96.52
4	26,101.80	93	12,206.42	43.82
6	39,152.70	62	16,828.72	25.57

Table 20. The industry and bi-objective GA degradation over B&B algorithm (%).

Machines per Stage	Industry		Bi-Objective GA	
	% E_{op}	% C_{max}	% E_{op}	% C_{max}
2	47.65	48.11	1.53	0.17
4	53.24	52.88	0.87	0.54
6	57.02	58.75	0.62	0.47

Figure 24 shows C_{max} considering working hours. We see that the results of the algorithm are significantly improved compared with industry. For two machines per stage, the difference is almost a halving of C_{max} from 186 to 96.52 h. In the case of four machines, it remains below half (93 and 43.82 h). In the case of six machines, the results are better by almost three times with regard to C_{max} (62 h versus 25.57 h).

Figure 25 shows E_{op} according to the processing time of machines (Table 3). We see that the B&B algorithm significantly improves upon the results obtained in industry. For two machines per stage, the E_{op} objective is reduced to almost half, and for four and six machines, the E_{op} objective of our algorithm is reduced to less than half.

Table 20 shows the percentage of degradation of the results of industry and those selected from the Pareto front obtained from the bi-objective GA compared with results of the B&B algorithm.

We observe that the degradation of E_{op} and C_{max} observed in industry are closer to or higher than 50%. Comparing B&B with our bi-objective GA, we observe that our results are less than 1.53% worse for E_{op} and 0.54% for C_{max} . Due to the fact that the B&B algorithm finds the global optimum, we demonstrate the quality of our algorithm.

8. Conclusions

Our main contributions are multifold:

- (1) We formulated the complex problem of the real-life industry environment of tortilla production considering two optimization criteria: total completion time and energy consumption;

- (2) We proposed a bi-objective solution to solve a hybrid flow shop with unrelated machines, setup time, and work in progress buffers. It is based on the known NSGA-II genetic algorithm;
- (3) We calibrated our algorithm using a statistical analysis of multifactorial variance. The ANOVA technique is used to understand the impact of different factors on the solution quality. A branch and bound algorithm was used to assert the obtained performance.
- (4) We provided a comprehensive experimental analysis of the proposed solution based on data from a tortilla production company in Mexico.
- (5) We demonstrated that our solutions are not more than 3.5 percent off the global optimum for both criteria. They save more than 48 percent of production time and 47 percent of energy consumption compared with the actual production plan followed by the company.

Author Contributions: All authors contributed to the analysis of the problem, designing algorithms, performing the experiments, analysis of data, and writing the paper.

Funding: This work is partially supported by Russian Foundation for Basic Research (RFBR), project No. 18-07-01224-a.

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

References

1. Linn, R.; Zhang, W. Hybrid flow shop scheduling: A survey. *Comput. Ind. Eng.* **1999**, *37*, 57–61. [[CrossRef](#)]
2. Quadt, D.; Kuhn, H. A taxonomy of flexible flow line scheduling procedures. *Eur. J. Oper. Res.* **2007**, *178*, 686–698. [[CrossRef](#)]
3. Yaurima, V.; Burtseva, L.; Tchernykh, A. Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Comput. Ind. Eng.* **2009**, *56*, 1452–1463. [[CrossRef](#)]
4. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [[CrossRef](#)]
5. Pei-Wei, T.; Jeng-Shyang, P.; Shyi-Ming, C.; Bin-Yih, L. Enhanced parallel cat swarm optimization based on the Taguchi method. *Expert Syst. Appl.* **2012**, *39*, 6309–6319.
6. Javanmardi, S.; Shojafar, M.; Amendola, D.; Cordeschi, N.; Liu, H.; Abraham, A. Hybrid Job Scheduling Algorithm for Cloud Computing Environment. In *Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014*; Komer, P., Abraham, A., Snasel, V., Eds.; Advances in Intelligent Systems and Computing Book Series; Springer: Berlin/Heidelberg, Germany, 2014; Volume 303, pp. 43–52.
7. Ribas, I.; Leisten, R.; Framiñan, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Comput. Oper. Res.* **2010**, *37*, 1439–1454. [[CrossRef](#)]
8. Luo, H.; Du, B.; Huang, G.; Chen, H.; Li, X. Hybrid flow shop scheduling considering machine electricity consumption cost. *Int. J. Prod. Econ.* **2013**, *146*, 423–439. [[CrossRef](#)]
9. Mori, M.; Fujishima, M.; Inamasu, Y.; Oda, Y. A study on energy efficiency improvement for machine tools. *CIRP Ann. Manuf. Technol.* **2011**, *60*, 145–148. [[CrossRef](#)]
10. Blum, C.; Chiong, R.; Clerc, M.; De Jong, K.; Michalewicz, Z.; Neri, F.; Weise, T. Evolutionary optimization. In *Variants of Evolutionary Algorithms for Real-World Applications*; Chiong, R., Weise, T., Michalewicz, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–29.
11. Liu, Y.; Dong, H.; Lohse, N.; Petrovic, S. Reducing environmental impact of production during a rolling blackout policy—A multi-objective schedule optimisation approach. *J. Clean. Prod.* **2015**, *102*, 418–427. [[CrossRef](#)]
12. Nilakantan, J.; Huang, G.; Ponnambalam, S. An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *J. Clean. Prod.* **2015**, *90*, 311–325. [[CrossRef](#)]
13. Wang, S.; Lu, X.; Li, X.; Li, W. A systematic approach of process planning and scheduling optimization for sustainable machining. *J. Clean. Prod.* **2015**, *87*, 914–929. [[CrossRef](#)]
14. Liu, C.; Huang, D. Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms. *Int. J. Prod. Res.* **2014**, *52*, 337–352. [[CrossRef](#)]

15. May, G.; Stahl, B.; Taisch, M.; Prabhu, V. Multi-objective genetic algorithm for energy-efficient job shop scheduling. *Int. J. Prod. Res.* **2015**, *53*, 7071–7089. [[CrossRef](#)]
16. Zhang, R.; Chiong, R. Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *J. Clean. Prod.* **2016**, *112*, 3361–3375. [[CrossRef](#)]
17. Mouzon, G.; Yildirim, M.B.; Twomey, J. Operational methods for minimization of energy consumption of manufacturing equipment. *Int. J. Prod. Res.* **2007**, *45*, 4247–4271. [[CrossRef](#)]
18. Dai, M.; Tang, D.; Giret, A.; Salido, M.A.; Li, W.D. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robot. Comput. Integr. Manuf.* **2013**, *29*, 418–429. [[CrossRef](#)]
19. Mansouri, S.A.; Aktas, E.; Besikci, U. Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *Eur. J. Oper. Res.* **2016**, *248*, 772–788. [[CrossRef](#)]
20. Hecker, F.T.; Hussein, W.B.; Paquet-Durand, O.; Hussein, M.A.; Becker, T. A case study on using evolutionary algorithms to optimize bakery production planning. *Expert Syst. Appl.* **2013**, *40*, 6837–6847. [[CrossRef](#)]
21. Hecker, F.T.; Stanke, M.; Becker, T.; Hitzmann, B. Application of a modified GA, ACO and a random search procedure to solve the production scheduling of a case study bakery. *Expert Syst. Appl.* **2014**, *41*, 5882–5891. [[CrossRef](#)]
22. Fonseca, C.M.; Fleming, P.J. An overview of evolutionary algorithms in multiobjective optimization. *Evol. Comput.* **1995**, *3*, 1–16. [[CrossRef](#)]
23. Hosseinabadi, A.A.R.; Siar, H.; Shamshirband, S.; Shojafar, M.; Nasir, M.H.N.M. Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in Small and Medium Enterprises. *Ann. Oper. Res.* **2015**, *229*, 451–474. [[CrossRef](#)]
24. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [[CrossRef](#)]
25. Jaimes, A.L.; Coello, C.A.C. Interactive Approaches Applied to Multiobjective Evolutionary Algorithms. In *Multicriteria Decision Aid and Artificial Intelligence*; Doumpos, M., Grigoroudis, E., Eds.; John Wiley & Sons, Ltd.: Chichester, UK, 2013. [[CrossRef](#)]
26. Lu, L.; Anderson-Cook, C.; Lin, D. Optimal designed experiments using a pareto front search for focused preference of multiple objectives. *Comput. Stat. Data Anal.* **2014**, *71*, 1178–1192. [[CrossRef](#)]
27. Wagner, T.; Trautmann, H. Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Trans. Evol. Comput.* **2010**, *14*, 688–701. [[CrossRef](#)]
28. Baker, K.R. *Introduction to Sequencing and Scheduling*; John Wiley & Sons, Ltd.: New York, NY, USA, 1974.
29. Pinedo, M. *Scheduling: Theory, Algorithms, and Systems*; Prentice Hall: Upper Saddle River, NJ, USA, 2002; 586p.
30. Abyaneh, S.H.; Zandieh, M. Bi-objective hybrid flow shop scheduling with sequence-dependent setup times and limited buffers. *Int. J. Adv. Manuf. Technol.* **2012**, *58*, 309–325. [[CrossRef](#)]
31. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Kan, A.H.G.R. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discret. Math.* **1979**, *5*, 287–326. [[CrossRef](#)]
32. Larrañaga, P.; Kuijpers, C.M.H.; Murga, R.H.; Inza, I.; Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artif. Intell. Rev.* **1999**, *13*, 129–170. [[CrossRef](#)]
33. Tan, K.C.; Lee, L.H.; Zhu, Q.L.; Ou, K. Heuristic methods for vehicle routing problem with time windows. *Artif. Intell. Eng.* **2001**, *15*, 281–295. [[CrossRef](#)]
34. Gog, A.; Chira, C. *Comparative Analysis of Recombination Operators in Genetic Algorithms for the Travelling Salesman Problem*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 10–17.
35. Hwang, H. An improved model for vehicle routing problem with time constraint based on genetic algorithm. *Comput. Ind. Eng.* **2002**, *42*, 361–369. [[CrossRef](#)]
36. Prins, C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. Oper. Res.* **2004**, *31*, 1985–2002. [[CrossRef](#)]
37. Starkweather, T.; McDaniel, S.; Mathias, K.E.; Whitley, C. A comparison of genetic sequencing operators. In *Proceedings of the 4th International Conference on Genetic Algorithms*; Belew, R.K., Booker, L.B., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1991; pp. 69–76.
38. The R Project for Statistical Computing. Available online: <https://www.r-project.org/> (accessed on 4 March 2017).

39. Montgomery, D.C.; Runger, G.C. *Applied Statistics and Probability for Engineers*; John Wiley & Sons, Inc.: New York, NY, USA, 1994.
40. Rahimi-Vahed, A.; Dangchi, M.; Rafiei, H.; Salimi, E. A novel hybrid multi-objective shuffled frog-leaping algorithm for a bi-criteria permutation flow shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2009**, *41*, 1227–1239. [[CrossRef](#)]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).