A Novel Heuristic to Minimize the Bottlenecks Presence in Batch Generation. Case of Study.

Pedro Henoc Ireta-Sánchez
Tecnológico Nacional de México/ Instituto Tecnológico de Saltillo
Elías Gabriel Carrum-Siller
Corporación Mexicana de Investigación en Materiales
David Salvador González-González
Corporación Mexicana de Investigación en Materiales
Ricardo Martínez-López (ricardo.ml@saltillo.tecnm.mx)
Tecnológico Nacional de México/ Instituto Tecnológico de Saltillo https://orcid.org/0000-0003-4828-0219

Research Article

Keywords: makespan, bottleneck, Simulated Annealing, Genetic Algorithm

Posted Date: June 18th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-559474/v1

A novel Heuristic to minimize the bottlenecks presence in batch generation. Case of study.

Pedro Henoc Ireta Sánchez[1], Elías Carrum Siller[2], David Salvador González González[2], Ricardo Martinez Lopez [1]

[1] Tecnológico Nacional de México/Instituto Tecnológico de Saltillo.
Blvd. Venustiano Carranza #2400, Col. Tecnológico, Saltillo, Coahuila, México C.P. 25280.

[2] Corporación Mexicana de Investigación en Materiales
Ciencia y Tecnología No. 790 Col. Saltillo 400, C.P. 25290. Saltillo, México

Corresponding author's E-mail: rmartinez@itsaltillo.edu.mx

ABSTRACT:

This paper presents a new heuristic method capable of minimizing the presence of bottlenecks generated when production batches have a distinct makespan. The proposed heuristic groups the jobs into items, where the one with the longest processing time in the batch determines the makespan. To test the heuristic, information was collected from a real paint process with two stations: one with a single cabin and the other with two parallel cabins. The capacity of processing jobs is limited by the cabin dimensions where jobs have different sizes and processing times. A makespan comparison between the heuristic proposed versus the First in First out (FIFO) dispatching rule that the case of study uses. Additionally, ten random instances based on data taken from the real process were created with the purpose to compare the new heuristic method versus Genetic Algorithm (GA) and Simulated Annealing (SA). The result of the comparison to FIFO, GA and SA showed that the proposed heuristic minimizes the bottleneck in a and creating batches almost with the same makespan. Results indicated a bottleneck time reduction of 96% when new heuristic method were compared to FIFO rule, while compared to Generic Algorithm and Simulated Annealing the bottleneck reduction were around 89% in both cases.

Keywords—makespan, bottleneck, Simulated Annealing, Genetic Algorithm

1. INTRODUCTION

The batch process is used in several manufacturing industries, such as chemical, food, pharmaceutical, metalwork, printed circuit board. This kind of processing rout is applied as a policy to reduce the cost of material handling, setups, processing times, idle time, and bottlenecks [1]. In the literature, this process is known as Batch Processing Machine (BPM) and it was introduced by Ikura and Gimple in 1986 where the authors applied in jobs with identical size [2]. Grouping jobs in batches is equivalent to the bin-packing problem that was demonstrated as an NP-Hard by Garey and Johnson in 1979 for n independent task, multiprocessor in parallel with the objective to minimize the makespan [3, 4]. The BPM has two versions: Serial batch (S-batch) and Parallel batch (P-batch). On S-batch the batches are processed in a serial form and the processing time is the summation of the processing times of all the jobs in the batch. In P-batch, jobs are processed at the same time as a batch on a machine or station in parallel, and the processing time is the longest time of the job in the batch [5, 6]. In BPM two important questions must be considered: 1) how to generate batches? and 2) how to sequence the batches to the process flow? To solve those questions, it is necessary to consider that jobs have a non-identical size (s_j) and different processing time (p_j), also that the machines have a limited capacity (C), and once the process begins, this cannot be stopped until it is over. Inter-batch job exchange is not allowed, and the total sizes of the jobs in the batch cannot exceed the machine capacity. All the jobs in the batch are processed and then released to be processed by the next stage [7, 8].

To solve BPM many methods based on the First Fit (FF) heuristic have been proposed and applied in different kind of production lines to minimize distinct objectives functions [9, 10]. Uzsoy [11], Ghazvini et al. [12], suggested heuristics for creating batches, their method was based on the bin packing problem for a single batch processing machine where the jobs had different sizes. Cheng et al. [13] established two algorithms applicable to jobs with multiple families and arbitrary sizes, additionally to that the machine had a limited capacity. Rafiee et al. [14] established three different methods to solve the problem based on a dynamic programming and compare them with Simplex and Branch & Bound; the authors used instances from Cheng et al. (2015), the result indicated that their methods find the optimal value in less time. Damodaran and Chang [15] presented two heuristics based on FF algorithm, applied them in parallel machines the results showed that the two heuristics performed better than Simplex method and SA. Arroyo and Leung [16] established three heuristics and compared them with Simplex in a problem when the jobs have different sizes and unequal ready times in an identical and unrelated parallel batch machine; the outcome of this

comparison indicated that the three heuristic outperform better than Simplex. Kaban et al [17] evaluated 44 dispatching rules in a simulation automotive processing line and compared with five different measures of performance where the Most Total Work Remaining (MTWR) rule obtained better results than the other dispatching rules. Sobeyko y Mönch [18] offered a solution for the flexible job shop with identical and unrelated parallel machines using a hybridized heuristic and compared with other heuristics, they used the Total Weighted Tardiness (TWT) as a measure of performance for 120 instances, the results indicated that they proposed is better on average in time to find the solution that reported by Brandimarte (1993). Nabli et al. [19] postulated the use of Local Search (LS) algorithm to solve a hybrid flexible flow shop with two stages, where the process has two machines in parallel at the beginning, and two dedicated machines at the second stage, the authors created random instances and compared with other dispatching rules, the results obtained for LS are close or equal to the optimal value and a low computational time.

With the development of mathematics, two important metaheuristics have been presented to solve de BPM when an exact method does not find the optimal value in a determinate time: Genetic Algorithm (GA) and Simulated Annealing (SA). For GA, Manjeshwar et al [20], they used GA to solve flow shop with to machines and contrast with SA and Simplex and generated their instances; in small instances GA obtained the same optimal value as Simplex, in medium and large instances GA reported outperformed in general better results than SA and in a few cases the same result. Driss et al. [21] proposed a New Genetic Algorithm (NGA), they used Brandimarte´s data and contrast with six algorithms, the solution obtained by NGA provides optimal results with a near than the data reported by Brandimarte, the scientists indicated that NGA was applied in a case of study where obtained a low makespan that the information from the company. Yan et al. [22] employed a multi-level approach using GA to optimize the makespan and the energy consumption at the same time and applied it in a company with a FJSP as a line production and the power demand from the different machines were collected from the literature, the experiment note that GA can assist to reduce the makespan and energy consumption. Yu et al. [23] used GA for a FJSSP, the authors compared their proposed with five start of arts metaheuristic selected previously, the results shows that their algorithm obtain a low relative percentage increase (RPI) in comparison with the five metaheuristics. Zhang et al. [24] improved the GA (IGA) to solve the FJSSP with multiple time constraints; IGA was compared with distinct different methods to generate the initial solution and used six instances, the investigation obtained that IGA results are better than the other methods. For SA, Damodaran et al. [25] established the use of SA in an identical parallel batch processing machines when the jobs have different processing time, size and the ready time is distinct to zero,

they compare SA with others metaheuristic using 200 instances with the lower bound has a measure of performance, the authors indicated that SA obtained values near to lower bound. Shahsavari-Pour and Ghasemishabankareh [26] used a hybrid algorithm with SA with GA to solve FJSSP with multiple objectives, the scientists used 2 instances and compared it with other algorithms, the hybrid establish a good effort to obtained a value near to the optimal solution than others methods. Song et al. [27] used an improved form of SA (ISA) in a flow shop scheduling problem; the journalist used a random data of 15 jobs with 10 machines and compared it with two SA variants; in the paper the results indicated that ISA obtained the same optimal value in a few minutes than the others variants. In Mirsanei et al. [28] the authors used a novel SA(NSA) to minimize the makespan in a hybrid flow shop with sequence-dependent set up times; the authors generated 2520 instances and compared with 2 metaheuristics and the Relative Percentage Deviation (RPD) was used as a performance measure; the values show that NSA got a lower value of RPD than the two metaheuristics.

The main objective of this study is to minimize the bottleneck in a paint line. The line has a single cabin where the jobs are cleaned, and two paint cabins in parallel. The process can be represented as a triplet: α, β, γ , where α is the machine characteristic, β is the job restriction and γ is the objective function [29] , so the line can be represented as: $FR_m \left| batch, s_j, p_j, d_j, C \right| Botcks$ where FR_m is a Flexible Job Shop with an Unrelated Parallel Machine, $batch$ means all the set of jobs, grouped together, s_j is the size of each job (quantity), p_j a processing time for each job, d dimensions of each job (length and width), C is the machine capacity, and $Botcks$ is the bottlenecks. The company where the experiment was made has a weekly production schedule with distinct types of jobs, and they use First In First Out (FIFO) as a dispatch rule. Each job has a distinct quantity of elements, different dimensions in square meters, processing time in minutes, and two distinct colors. In general, the process begins when the programmer decides to take the first job in the schedule and send it to the washing cabin to remove burrs. Once the process is finished, the second job is moved to the washing cabin, and so on. The problem begins when the scheduler decides to submit the first job as it appears on the production schedule, without considering the number of elements or their processing times. Once the job arrives at the wash station, the operator enters as many elements as possible in the cabin and leaves the others outside. When the process has finished, they immediately send the elements to the paint area, without waiting for the remaining elements. Only on a few occasions, a job can be entered entirely into the wash cabin and when the job is finished, it is sent completely to the painting station. This procedure type generates a bottleneck between the elements that enter the washing cabin and those that are waiting to process on the next stage, in addition to

the fact that a cabin stay empty due to elements are sent immediately to the painting lines. This type of operation causes bottlenecks between jobs. As a result, the company have a high penalty per minute of bottleneck generated.

To solve this problem this study, propose a new Heuristic to form batches with the same processing time. The heuristic will create a batch with the elements of the job that have the longest processing time and dimension. The processing time for the batch will be the time for that element, and the rest of the elements of the other jobs will be entered randomly, taking into count the restriction that the sum of their times should not be greater than the processing time of the batch. To verify the effectiveness of Heuristic, information from the company was gathered and compared to FIFO dispatching rule. Also, 10 instances were generated using that information and comparing it to GA and SA.

Finally, this paper is organized as follows: Section 2 describes the application of different heuristics and metaheuristics in a batch processing machine as found in the literature. Section 3 presents the mathematical model of the addressed problem. In section 4, the heuristic proposed. Section 5 describes the computational results. And lastly, in section 6, conclusions and future work.

2. MATHEMATICAL MODEL

The problem under study can be defined as follows: a set of jobs $j \in J$, in which each job has a total processing time p_j and a size s_j. The jobs must be grouped in batches b, and each job can be divided by elements. One single cabin or machine in the first station and an arrange of unidentical parallel machines $m \in M$ or cabins in the second station. Each cabin has a different capacity C. The batch processing time P_b is equal to the longest processing time of items of the job with the longest processing time and to the number of items. The aim is to minimize the presence of bottlenecks that are generated between batches because they have an unequal makespan.

The main assumptions of the addressed problem are:

1. The capacity of the buffer between stages is unlimited.
2. Each job can be divided into elements.
3. Every element has a size and a processing time.
4. Every batch can be formed with distinct elements.
5. Skipping a station is not permitted.
6. The dimension of each batch is equal to the sum of the dimensions of each element in the batch.

7. The processing time for each cabin depends on the processing time of each batch.
8. Elements from a batch are allowed to be outside of the cabin if the total sum of the dimensions of the elements exceeds the machine´s capacity.
9. It is not allowed to send elements from a batch separately to the next workstation.
10. Batches are grouped into runs and the processing time must be less or equal to the time of the work shift.

The follow notations are presented to describe the problem:

$J = index\ for\ job\ (J = 1, \dots to\ N)$

$m = index\ for\ machine\ or\ cabine\ \ (m = 1, \dots to\ M)$

$E_j = index\ for\ Elements\ from\ jobs\ (j = 1, \dots to\ E), E_j \in J$

$J_{max} = job\ with\ the\ longest\ processing\ time, (J_{max} \in J)$

$E_j^{max} = element\ of\ \ J_{max}.$

$C = capacity\ in\ dimension\ (square\ meters)\ for\ each\ cabin\ or\ machine.$

$S_j = job\ size\ in\ square\ meters$

$E_s = size\ for\ each\ element\ from\ job, size\ in\ square\ meters, (E_s \in\ S_j).$

$p_j = processing\ time\ for\ each\ job\ in\ minutes$

$E_p = processing\ time\ for\ each\ element\ from\ job, time\ in\ minutes, (E_p \in\ p_j)$

$B = set\ of\ batches$

$b = b \in B$

$b_t = processing\ time\ for\ the\ batch\ and\ is\ equal\ to\ E_j^{max}$

$b_s = batch\ dimension\ in\ square\ meters$

$T_t = time\ of\ the\ work\ shift$

$C_b^{max} =\ is\ the\ makespan\ for\ each\ batch, (C_b^{max} = E_j^{max}).$

$R = set\ of\ runes$

$r = r \in R\ (runs)$

$r = runs\ grouped\ in\ batches\ to\ be\ processing\ in\ the\ time\ that\ the\ work\ shift\ last.$

$X_{j,b} = \{1\ \ if\ the\ job\ j\ is\ asigned\ to\ batch\ b;\ O\ otherwise.$

$X_{E_j,b,m} = \{1\ \ the\ element\ \ j\ is\ asigned\ to\ batch\ and\ processed\ in\ machine\ m; 0\ otherwise.$

$\{b_1, b_2, ., b_{n-1}\} = \{1\ \ if\ batch\ b\ is\ grouped\ in\ runs;\ 0, otherwise.$

$(r_l - r_{l+1}) = \{Negative\ value\ means\ bottleneck.$

Below the mathematical model is presented based on the assumptions mentioned above.

The aim objective in this problem is to minimize the bottleneck between batches that is represented in equation 1.

$$Min\ Bottleneck = Min\ Dif\ (r_l - r_{l+1}) \quad (1)$$

Subject to:

$$b_s \leq C \quad (2)$$

$$\sum_{j=1}^{n} \sum_{b=1}^{m} X_{j,b} = 1, j = 1, \dots, n, b = 1 \dots, m \quad (3)$$

$$\sum_{E_j=1}^{n} \sum_{b=1}^{m} X_{E_j,b,m} = 1, j = 1, \dots, n, b = 1 \dots, m \quad (4)$$

$$E_j^{max} \leq J_{max} \quad (5)$$

$$\sum_{i=1}^{n-1} b_i = \{b_1, b_{2,..,} b_{n-1}\};\ \{b_1, b_{2,..,} b_{n-1}\} = 1 \quad (6)$$

$$\sum_{b=1}^{n-1} |b \in r| \leq T_t \quad (7)$$

$$C_b^{max}, X_{E_{j,b,m}}, X_{j,b} \geq 1 \quad (8)$$

Eq. 1 is the objective function. Restrictions (2) and (3) means to the batch dimension must be less or equal to the dimension of each cabin and indicates that all jobs must be in a batch, respectively. Restriction (4) mentions that all the elements of the jobs must be in a batch and processed in a cabin. Restriction (5) represents that the element with the largest processing time in a batch must be less or equal than the processing time of the job. Restriction (6) and (7) show that the batches formed must be grouped into runs and the sum of the makespan of each batch must be less. Finally, restriction (8) is the non-negative value.

3. HEURISTIC

Batch problems can be classified in two ways according to processing times: 1) a constant batch processing time, where the processing times are independent of these jobs and 2) variable processing time, where it depends on the jobs that are in the batch, as Damodaran et. al. mentioned [8]. To solve the study problem, the following decisions need to be made: a) how to generate batches with the same makespan and b) how to reduce the bottleneck.

Derived from above, this Heuristic creates batches with equal makespan promoting beneficial effects on the bottleneck reduction. This heuristic divides each job into elements and the makespan is the processing time of the element with the longest processing time, the processing time of the

rest of the elements in the batch cannot exceed that time. Subsequently, the Heuristic groups the batches into runs in order to minimize the bottleneck.

The pseudo code for the Heuristic is shown below:

BEGIN

1. Separate the job (N) with the longest processing time (P_N) and size from the schedule and divide it into the number of items (N_i).
2. The jobs that were not selected on step 1 (O) they will be divide into element (O_i) and have a processing time (P_O) . Each element must be identified with an integer number starting with the number 1 for the first ordered element, then number 2, and so on.

Star

3. Permute (O). The length of the permutation should be equal to the number of items in the remaining jobs that were not selected in step 1.

End

Star

For 1 to N_i. Generate batches

If $P_{Oi} \leq P_{Ni}$

4. *Assign (O_i) start with the first element of the permutation in N*

Else

Create a new batch with the next N_i

Note: If exist items from O that have not been entered into the batches, new batches must be formed using the next job with the longest processing time and the sum of the processing times of the items that are making up the batch should not exceed the processing time of that element.

End

End

5. Sort batches according to their processing time (use SPT rule) and grouped them into runs use FIFO until the sum of the makespan of each batch are less or equal than the duration of the shift work.
6. Calculate the bottleneck between runs using equation (9) and sum all negative values.
$$(X_t - X_{t+1}) \quad (9)$$

Note: If the result of $(X_t - X_{t+1})$ is a negative value it means that the job X_{t+1} is still in process in the machine and the value obtained indicates the time is delay all the batches.

7. Identify the largest bottleneck of the results of the predecessor minus the successor value (this means that the predecessor value is smaller than the successor)
8. Swap elements of the runs that generate bottlenecks to the run of the previous step until the total sum of the bottleneck is less than the first obtained.
9. In case to exist machines in parallel, each run should be divided into equal parts according to the number of parallel machines.

End

4. APPLICATION AND RESULTS

To test the Heuristic, information was collected from the case study: two production schedules containing the job dimensions in square meters, processing time in minutes and quantity of elements per job to produce, and also the dimension in square meters of each cabin. The experiment consisted of two parts. In the first one, the collected information was used to create the batches. In the second part, ten random instances were stablished using the collected information. The objective is to create batches with an equal processing time.

Figure 1 shows the process flow from the paint line and Table 1 shows the job information collected. The first column is the job number and color of the production schedule. The second and third columns are the dimension in square meters and the processing time per minute, respectively. Finally, Column 4 is the number per element.

Fig. 1. *Process flow*

Table 1*. Data collected.*

Color 1	**Dimension**	**Processing time**	**Quantity (size)**
J1	13.86	165.73	15
J2	7.88	33.94	15
J3	11.154	16.01	10
J4	3.473	96.91	20
J5	1.397	45.5	8
J6	1.027	6.09	1
Color 2	**Dimension**	**Processing time**	**Quantity (size)**
J7	3.726	2.05	60
J8	90.684	50.92	20
J9	16.382	10.23	5
J10	11.154	16.01	5
J11	3.841	1.69	6

Table 2 shows the results of the way that the company sequences the jobs to be processed in the production line as a batch. For color 1, there are two batches of job 6 with different processing times, and the rest of the batches are made up of each job. This was the result of applying the methodology in which the company carries out the process through the FIFO rules, placing as many jobs as possible into the wash cabin in the first job while leaving the rest of the elements of that job outside the station until the process is finished. Immediately after, they send the elements to the paint station to be processed in any cabin, for that reason the sum of the makespan of the batches and the

bottleneck are large. On the other hand, for color 2, it is observed that there are 20 batches of job 11. Since the wash station can only process one element, the bottleneck is short due to the makespan values for almost all the batches are very close.

***Table 2**. Processing time obtained*

Color 1	**Elements per batch**	**Makespan per Batch**
1	9J1	1491.57
2	6J1	994.38
3	15J2	509.1
4	10J3	160.1
5	20J4	1938.2
6	8J5	364
7	1J6	1.027
Color 2	**Elements per batch**	**Makespan per batch**
1	33J7	67.5
2	27J7	55.35
3 TO 22	1J8	50.92
23	5J9	51.15
24	5J10	80.05
25	6J11	10.14

For color 1, the total sum of the makespan is 5458.37 minutes and the bottleneck is -1778.1. For color 2, the total makespan is 315.11 and the bottleneck is -29.13 minutes.

To prove the Heuristic, the same information was used. As mentioned before, the Heuristic has two parts: the first part is to form batches, and the second one groups batches into runs to minimize the bottleneck. Table 3 shows the results obtained from the first part. The first column is the batch number, the second column shows the elements in the batch and the third column is the processing time of the batch.

***Table 3.** Results from Heuristic before to group the batches into runs.*

Batch number (Color 1)	**Makespan**
1 TO 15	165.75
16	45.5
17 TO 20	96.91
21	45.5

22 TO 24	96.91
25	45.5
Batch number (Color 2)	**Makespan**
1 TO 20	50.90

As it is shown in table 3, the batches formed with the color 1 had, in general, a constant makespan but when calculating the bottleneck, the result is -102.82 minutes. For color 2, all the batches have the same makespan, therefore, no presence of bottleneck. The result of the makespan of each batch is the processing time of the element in the batch.

For the second part, the STP rule was used to group batches to create runs with the same makespan and minimize the bottleneck. A swap of elements was added. Four elements: three jobs j3 from run 1, and one j3 from run 2 for color 1. The bottleneck obtained was -60.96 minutes, with almost all the runs having the same makespan. For color 2, there was no bottleneck because all the batches have the same makespan. Figure 2 shows the results of grouping the batches in runs. For the color 1 almost all the runs have the same makespan because the runs were grouped in pairs.
After all, the sum of the makespan does not exceed the time that the work shift lasts. For the color 2, they were also grouped in pairs to have the same number of runs as the color 1 ones.

Fig. 2*. Group of batches in runs.*

As mentioned before, the study problem has two paint cabins and, in this case, each run was divided according to the number of parallel machines. Table 4 indicates the results of the bottleneck.

Table 4. *Bottleneck obtained for the paint cabin.*

	FIFO			**Heuristic**	
Batches	Cabin 1	Cabin 2	Runs	Cabin 1	Cabin 2
Color 1	-889.05	-889.05	Color 1	-32.87	-28.09
Color 2	-33.41	-32.02	Color 2	0	0

When the Heuristic is compared to FIFO, it minimizes the bottleneck but generates more batches than FIFO because it creates a batch for each element of the job that has a larger size and processing time.

In the second part of the experiment, 10 random instances were created, according to the parameters that Talliard [30] established, using the information from the case study. The length of the shift work is 420 minutes. The highest value of each parameter from scheduling was taken as an upper value. Factor and levels are shown in table 5.

Table 5. *Parameters, factors and level*s.

Parameter	Factor	Level
Number of jobs	n	Uniform $[1, n]$
Job size	s_j	Uniform $[1, s_j]$
Job processing time	p_j	Uniform $[1, p_j]$

The main characteristics of these instances are the following:

- Scheduling with different jobs.
- Jobs with distinct elements.
- Elements with individual dimension and processing time.
- The processing time of each element is given in minutes.
- The dimension of each element is given in square meters.

To test the Heuristic in comparison to Simulated Annealing and Genetic Algorithm, both metaheuristics were run 5 times. For both metaheuristics, the operating method proposed by Damodaran et al [25] for SA and [31] for GA. For the generation of batches, the development by Bertisimas was used. [32]. From each run generated, the batch with the longest processing time was selected to be compared to the others. The batch with the shortest time is the one that determined which run to use to compare to Heuristice. The software used was a Mathematical software in a Laptop Sony Vaio E Series with an Intel Core I5, CPU 250 GHz, and 64GB of RAM.

Table 6 shows the results of the comparison between the Heuristic and the metaheuristics. The first row is the name of the algorithm and the second row is the bottleneck generated. It can be observed that proposed Heuristic resulted in a lower bottleneck in comparison to GA and SA. This result in some instances had jobs whose elements formed all the batches, so for this reason they obtained a bottleneck equal to 0 or very close to this value. It can also be noticed that high bottleneck is obtained, it means that some elements that were not part of the batches, formed were grouped with others with different times. In the case of the values obtained by both metaheuristics, they generated batches with elements in a random way, taking as a restriction that the sum of the dimensions of the elements should not exceed the dimension of the first cabin. Once the batches were formed, the processing times of the elements were added, resulting in batches with different makespan and a high bottleneck.

Table 6. *Comparison between the Heuristic with SA and GA.*

Instance 1				**Instance 2**			
Algorithm	SA	GA	Heuristic	Algorithm	SA	GA	Heuristic
Bottleneck	-980	-971	-125	Bottleneck	-287	-406	-17

Instance 3				Instance 4			
Algorithm	SA	GA	Heuristic	Algorithm	SA	GA	Heuristic
Bottleneck	-2277	-1984	0	Bottleneck	-1594	-1455	-110
Instance 5				**Instance 6**			
Algorithm	SA	GA	Heuristic	Algorithm	SA	GA	Heuristic
Bottleneck	-2613	-2800	-302	Bottleneck	-1216	-1349	-30
Instance 7				**Instance 8**			
Algorithm	SA	GA	Heuristic	Algorithm	SA	GA	Heuristic
Bottleneck	-214	-386	-74	Bottleneck	-1882	-1603	-240
Instance 9				**Instance 10**			
Algorithm	SA	GA	Heuristic	Algorithm	SA	GA	Heuristic
Bottleneck	-837	-769	-261	Bottleneck	-396	-497	-122

Figure 3 compares the makespan obtained for each instance used by the proposed heuristic, GA, and SA. The proposal obtained a low value than SA and GA because the proposal generates batches according to the processing time of the element from the job with the highest total processing time as quantity and when grouping them into runs the sum of the times must not exceed the duration of the work shift, otherwise, SA and GA generate batches taking as a restriction that the sum of the dimensions does not exceed the capacity of the cabin, resulting batches with different processing times.

Fig. 3. *Comparison between Heuristic, GA and SA*

For parallel painting, each run for Heuristic or batch for SA or GA was divided into parts according to the number of cabins. When compared, it can be seen that the Heuristic generated less bottleneck than SA and GA. Table 7 shows the bottleneck generated by each batch by the instances when they are processed in the paint cabins.

Table 7. *Bottleneck results in paint cabins.*

Cabine 1				Cabine 2			
Instance number	GA	SA	Heuristic	Instance number	GA	SA	Heuristic
1	-266	-252	-256	**1**	-772	-443	-381
2	-223	-152	-30	**2**	-175	-167	0
3	-223	-152	0	**3**	-152	-167	0
4	-818	-1066	-160	**4**	-696	-590	-110
5	-2552	-2616	-159	**5**	-580	-285	-81
6	-1444	-873	-227	**6**	-578	-470	-232
7	-224	-128	-23	**7**	-162	-117	-51
8	-1148	-1000	-343	**8**	-901	-1222	-289
9	-333	-503	-188	**9**	-491	-381	-123
10	-348	-187	0	**10**	-285	-204	0

In general, the Heuristic obtained lower bottleneck time than SA and GA in the parallel cabins. When the Heuristic obtained a bottleneck equal to 0 for both cabins, indicates that each run contains batches with same processing time; on the other hand, when bottlenecks exist, it is due to the runs have batches with different processing time. In the case of GA and SA, it was not possible to divide the batches into equal parts, as mentioned before, the way to generate the batches depends on the size of the washing cabin, so when attempting to divide the batch to be processed at the same time, considering the paint cabins, it was not possible to have equal processing times.

5. CONCLUSIONS AND FUTURE WORK

The proposed heuristic creates batches using the elements of the job that had the longest processing time and batch dimensions in order to minimize the bottleneck. First, the principal objective was to evaluate the heuristic using the information from a real case study to find if the proposed method generates lower processing times than FIFO dispatching rule. Results showed that when batches are grouped in runs the heuristic reduces the bottleneck in painting line in a 96.94%.

On the other hand, ten random instances were generated to verify the efficiency of the heuristic when compared to SA and GA metaheuristics. In some instances, the heuristic registered higher processing times than metaheuristics mentioned, but if batches were grouped in runs, the bottlenecks were reduced in 89.55% comparison with SA and GA.

The heuristic when generating batches with the same processing time can apply in a single machine or machines in series; for parallel machines, it is possible to use if the runs can be separated into batches with the same processing time.

In conclusion, the heuristic minimizes the value of bottleneck than FIFO rule, SA and GA, when runs with equal processing time are generated and when applied in simple or serial machines. In parallel machines, the heuristic does not create a bottleneck if the runs can be divided with the same processing time. In parallel machines, the heuristic does not create a bottleneck when dividing the runs into the number of batches that form it because they have the same processing time and each batch must have an element from the job with the longest makespan.

Finally, as future work, the Heuristic will be tested in dynamic environments such as arrival of news jobs, machine failure, job priority changes, etcetera.

6. ACKNOWLEDGE

The author wishes to thank to the Tecnológico Nacional de México/Instituto Tecnológico de Saltillo for the support during the doctoral study. Also, to the National Council of Science and Technology (CONACYT) for the scholarship granted.

Originality: The authors declare that the article is original and never has presented in another journal.

Funding: Not apply.

Economic Interests: The authors declare that they did not have financial or proprietary interest in any material from this article.

Conflict of interest: The authors declare that they have not conflict of interest.

Data availability: the data of the instances are available upon request to the author.

Code availability: Not apply.

Authors contributions: Pedro Henoc Ireta Sánchez was the responsible for the writing of the article, the proposed Heuristic presented to solve the case of study as the instances, compilation, and presentation of results. Ricardo Martínez López was responsible for the direct tutorial and review of the writing of the article. David S. González-González was responsible for the final review of the article. Finally, Elias Gabriel Carrum Siller was the responsible for the providing the information from the case of study.

REFERENCES

[1] R. Xu, H. Chen y X. Li , «Makespan minimization on single batch-processing machine via ant colony optimization,» *Computers & Operations Research,* vol. 39, pp. 582-593. https://doi.org/10.1016/j.cor.2011.05.011, 2012.

[2] Y. Ikura y M. Gimple, «Efficient scheduling algorithms for a single batch processing machine, » *Operations Research Letters,* vol. 5, pp. 61-65. https://doi.org/10.1016/0167-6377(86)90104-5, 1986.

[3] E. Coffman, M. Garey y D. Johnson, «An applications of bin-packing to multiprocessor scheduling, » *SIAM Journal fo Computation,* vol. 7, pp. 1-17. https://doi.org/10.1137/0207001, 1978.

[4] J. E. C. Arroyo, J. Y.-T. L. Leung y R. G. Tavares, «An iterated greedy algorithm for total flow time minimization in unrelated parallel batch machines with unequal job release times,» *Engineering Applications of Artificial Intelligence,* vol. 77, pp. 239-254.https://doi.org/10.1016/j.engappai.2018.10.012, 2019.

[5] Z.-h. Jia, W. Chao y J. Y.-T. Leung, «An ACO algorithm for makespan minimization in parallel batch machines with non-identical job size and incompatible job families, » *Applied Soft Computing,* vol. 38, pp. 395-404.https://doi.org/10.1016/j.asoc.2015.09.056, 2016.

[6] M. Mathirajan y A. Sivakumar, «A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, » *International Journal of Advanced*

Manufacturing Technology, vol. 29, pp. 990-1001.https://doi.org/10.1007/s00170-005-2585-1, 2006.

[7] Z.-h. Jia, K. Li y J. Y. Leung, «Effective heuristic for makespan minization in parallel batch machines with non-identical capacities, » *International Journal of Production Economics,* vol. 169, pp. 1-10. https://doi.org/10.1016/j.ijpe.2015.07.021, 2015.

[8] P. Damodaran, O. Ghrayeb y M. C. Guttikonda, «GRASP to minimize makespan for a capacitated batch-processing machine, » *International Journal of Advanced Manufacturing Technology,* vol. 68, pp. 407-414.https://doi.org/10.1007/s00170-013-4737-z, 2013.

[9] A. Costa, F. A. Cappadonna y S. Fichera, «A novel genetic algorithm for the hybrid flow shop scheduling with parallel batching and eligibility constraints, » *The International Journal of Advanced Manufacturing Technology,* vol. 75, pp. 833–847. https://doi.org/10.1007/s00170-014-6195-7, 2014.

[10] M. Rojas-Santiago, P. Damodaran, S. Muthuswamy y M. C. Vélez-Gallego, «Makespan minimization in a job shop with a BPM using simulated annealing, » *International Journal Advanced Manufacturing Technology,* vol. 68, pp. 2383-2391.https://doi.org/10.1007/s00170-013-4858-4, 2013.

[11] R. Uzsoy, «Scheduling a single batch processing machine with non-identical job sizes,» *International Journal of Production Research,* vol. 32, pp. 1615-1635.https://doi.org/10.1080/00207549408957026, 1994.

[12] F. J. Ghazvini y L. Dupont, «Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes,» *International Journal of Production Economics,* vol. 55, pp. 273-280.https://doi.org/10.1016/S0925-5273(98)00067-X, 1998.

[13] B. Cheng, J. Cai, S. Yang y X. Hu, «Algorithms for scheduling incompatible job families on single batching machine with limite capacity, » *Computers and Industrial Engineering,* vol. 75, pp. 116-120.https://doi.org/10.1016/j.cie.2014.06.014, 2014.

[14] N. Rafiee Parsa, B. Karimi y S. Moattar Husseini, «Exact and heuristic algorithms for the just-in-time scheduling problem in a batch processing system,» *Computers and Operations Research,* vol. 80, pp. 173-183.https://doi.org/10.1016/j.cor.2016.12.001, 2017.

[15] P. Damodaran y P.-Y. Chang, «Heuristic to minimize makespan of parallel batch processing machines,» *International Journal of Advanced Manufacturing Technology,* vol. 37, pp. 1005-1013.https://doi.org/10.1007/s00170-007-1042-8, 2008.

[16] J. E. C. Arroyo y J. Y.-T. Leung, «Scheduling unrelated parallel batch processing machines with non-identical job size and equal ready times,» *Computers and Operations Research,* vol. 78, pp. 117-128.https://doi.org/10.1016/j.cor.2016.08.015, 2017.

[17] A. K. Kaban, Z. Othman y D. S. Rohmah, «Comparison of dispatching rules in job-shop scheduling problem using simulation: case of study,» *International Journal of Simulation Modelling,* vol. 3, pp. 129-140.https://doi.org./10.2507/IJSIMM11(3)2.201, 2012.

[18] O. Sobeyko y L. Mönch, «Heuristics approaches for scheduling jobs in large-scale flexible job shop,» *Computers and Operations Research,* vol. 68, pp. 97-109.https://doi.org/10.1016/j.cor.2015.11.004, 2016.

[19] Z. Nabli, S. Khalfallah y O. Korbaa, «Heuristics for the Hybrid Flow Shop Scheduling Problem with Parallel Machines at the First Stage and Two Dedicated Machines at the Secong Stage,» *Industrial Engineering and Management Systems,* vol. 14, pp. 22-31.https://doi.org/10.1007/978-3-319-76348-4_67, 2015.

[20] P. K. Manjeshwar, P. Damodaran y . K. Srihari, «Genetic algorithms for minimizing makespan in a flow shop with two capacitated batch processing machines,» International Journal of Advanced Manufacturing Technology, vol. 55, pp. 9-12. https://doi.org/10.1007/s00170-010-3150-0, 2011.

[21] I. Driss, K. N. Mouss y A. Laggoun, «A new genetic algorithm for the flexible job-shop scheduling problems,» *Journal of Mechanical Science and Technology,* vol. 29, pp. 1273-1281.https://doi.org/10.1007/s12206-015-0242-7, 2015.

[22] J. Yan, L. Li, F. Zhao, F. Zhang y Q. Zaho, «A multi-level optimization approach for energy-efficient flexible shop scheduling,» *Journal of Cleaner Production,* vol. 137, pp. 1543-1552.https://doi.org/10.1016/j.jclepro.2016.06.161, 2016.

[23] C. Yu, Q. Semeraro y A. Matta, «A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility,» *Computers and Operations Research,* vol. 100, pp. 211-229.https://doi.org/10.1016/j.cor.2018.07.025, 2018.

[24] G. Zhang, Y. Hu, J. Sun y W. Zhang, «An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints,» *Swarm and Evolutionary Computation,,* vol. 54, p. 100664.https://doi.org/10.1016/j.swevo.2020.100664, 2020.

[25] P. Damodaran y M. C. Vélez-Gallego, «A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times,» *Expert Systems with Applications,* vol. 39, pp. 1451-1458.https://doi.org/10.1016/j.eswa.2011.08.029, 2012.

[26] N. Shahsavari-Pou y B. Ghasemishabankareh, «A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling,» *Journal of Manufacturing Systems,* vol. 32, pp. 771-780.https://doi.org/10.1016/j.jmsy.2013.04.015, 2013.

[27] S. Song , J. Ren y J. Fan , «Improved Simulated Annealing Algorithm Used for Job Shop Scheduling Problems,» *Advances in Electrical Engineering and Automation,* vol. 139, pp. 17-25.https://doi.org/10.1007/978-3-642-27951-5_3, 2012.

[28] H. S. Mirsanei, M. Zandieh y M. J. Moayed, «A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times,» *Journal of Intelligent Manufacturing,* vol. 22, pp. 965–978.https://doi.org/10.1007/s10845-009-0373-8, 2011.

[29] R. L. Graham, E. L. Lawler, J. K. Lenstra y A. H. RinnooyKan, «Optimization and Approximation in Deterministic Sequencing and Scheduling: a SURVEY,» *Annals of Discrete Mathematics,* vol. 5, pp. 287-326.https://doi.org/10.1016/S0167-5060(08)70356-X, 1979.

[30] E. Talliard, «Benchmarks for basic scheduling problems,» *European Journal of Operational Research,* nº 64, pp. 278-285.https://doi.org/10.1016/0377-2217(93)90182-M, 1993.

[31] P. Damodaran, P. Kumar Manjeshwar y K. Srihari, «Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms,» *International Journal of Production Economics,* vol. 103, p. 882–891.https://doi.org/10.1016/j.ijpe.2006.02.010, 2006.

[32] D. J. Bertsimas , «A Vehicle Routing Problem with Stochastic Demand,» *Operations Research,* vol. 40, pp. 574-585.https://doi.org/10.1287/opre.40.3.574, 1992.

Figures

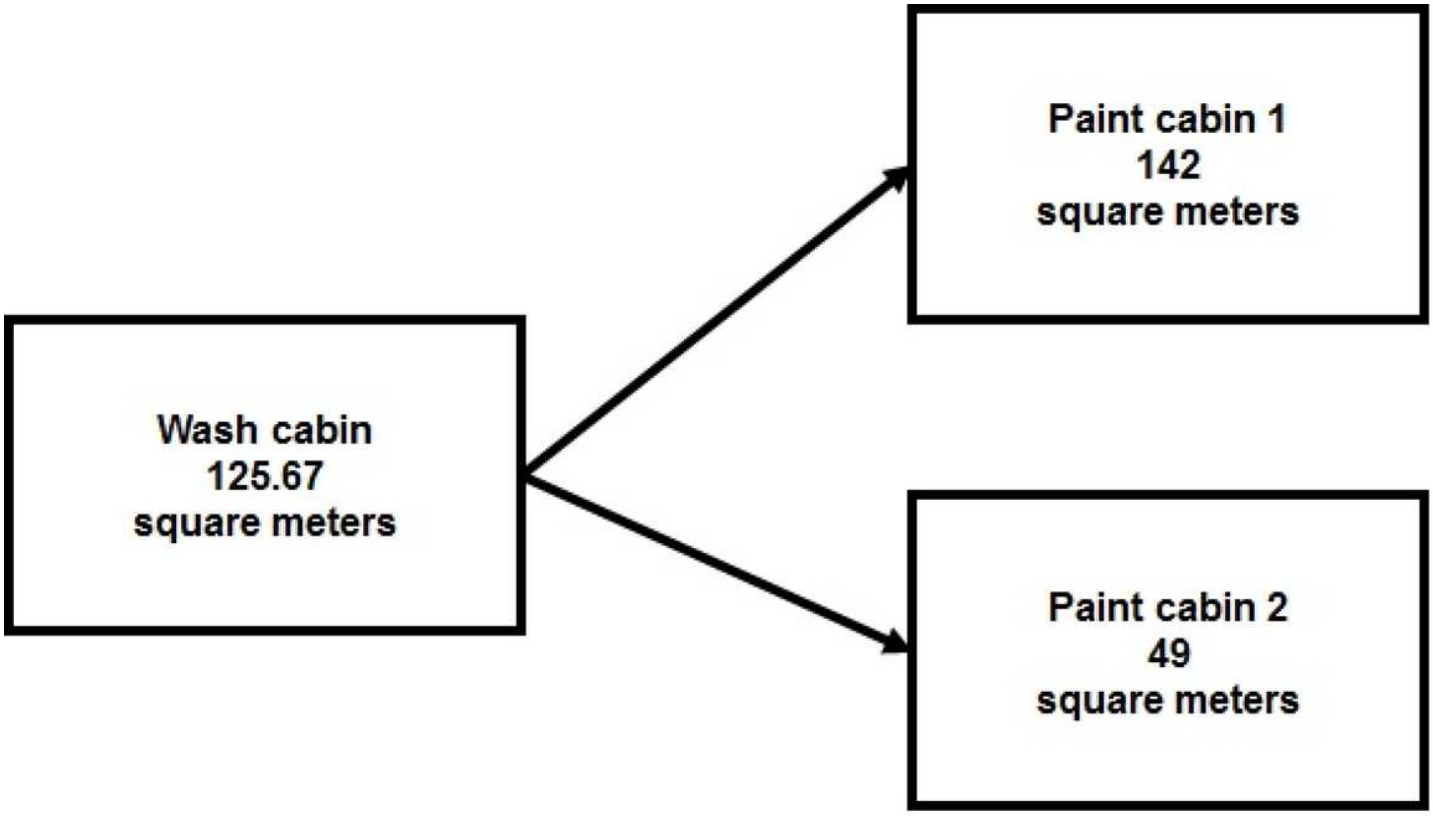


Figure 1

Process flow

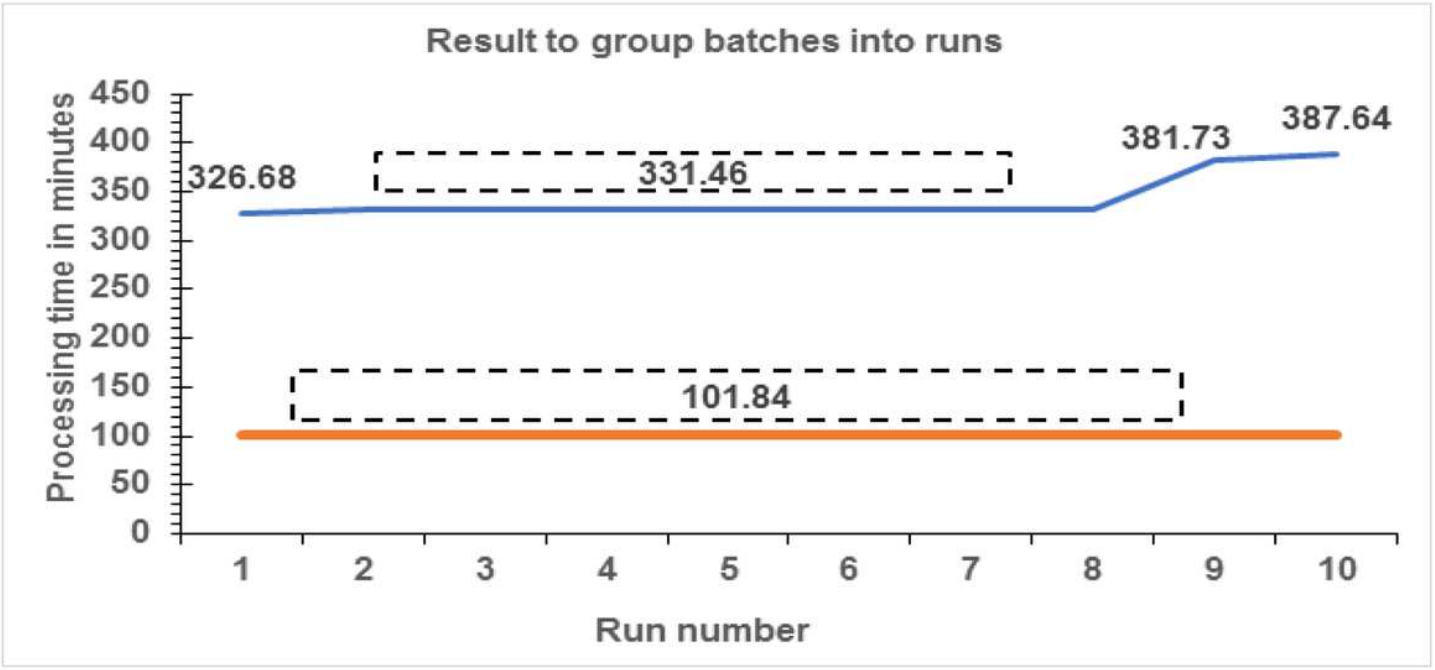
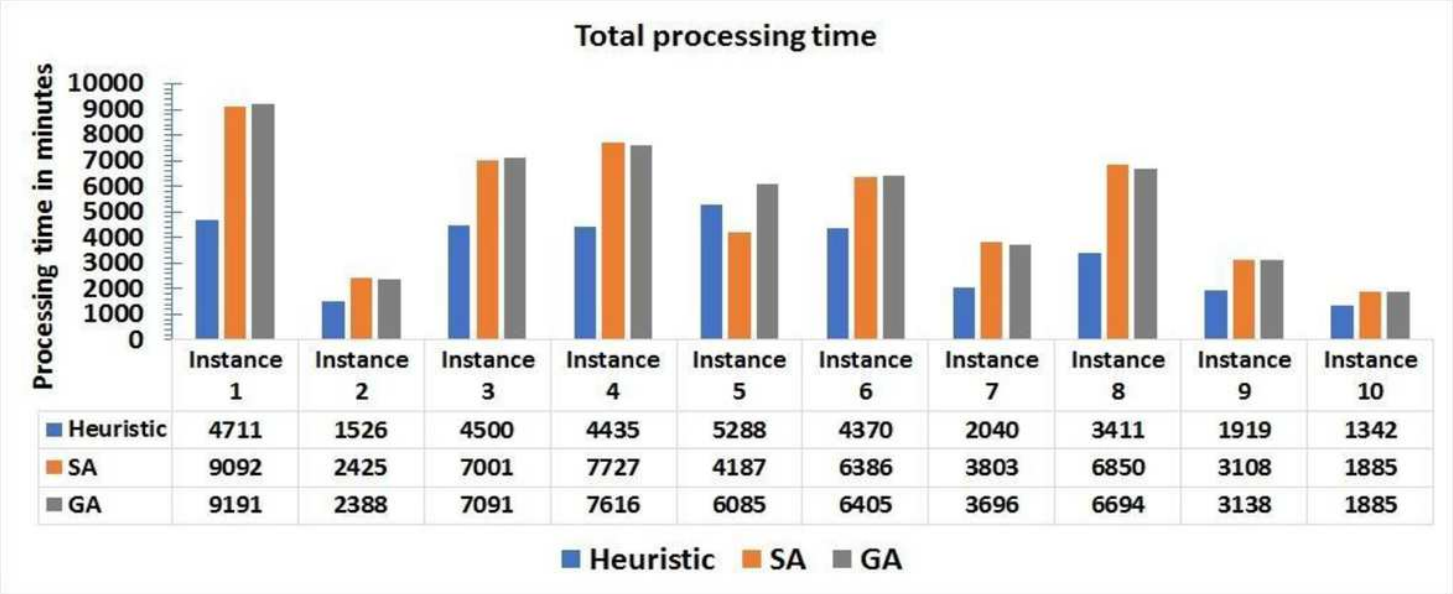


Figure 2

Group of batches in runs



	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6	Instance 7	Instance 8	Instance 9	Instance 10
Heuristic	4711	1526	4500	4435	5288	4370	2040	3411	1919	1342
SA	9092	2425	7001	7727	4187	6386	3803	6850	3108	1885
GA	9191	2388	7091	7616	6085	6405	3696	6694	3138	1885

Figure 3

Comparison between Heuristic, GA and SA