25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

COB-2019-1761
GENETIC ALGORITHM TO PROVIDE A SOLUTION IN ORDER TO MINIMIZE MAKESPAN AND FLOW TIME IN A FLOW SHOP WITH BLOCKING ENVIRONMENT

Pedro Eduardo Hernandes Natal, pedroehn@hotmail.com[1]
Mauricio Iwama Takano, takano@utfpr.edu.br[1]
Edson Hideki Koroishi, edsonh@utfpr.edu.br[1]
[1]Universidade Tecnológica Federal do Paraná campus Cornélio Procópio, Av. Alberto Carazzai, 1640 – Cornélio Procópio – PR. CEP 86300-000,

Abstract: *In this paper, a genetic algorithm for a flow shop problem considering m-machines, n-jobs and zero buffer environment with the objective of minimizing makespan and flow time is proposed. The objective of this paper is to find the best size of the initial population (PS) that provides good quality solutions keeping low computational time. According to (ZINI,2009) Genetic algorithm is applicable for a wide range of problems and it features good performances, it doesn't use just local information, therefore it doesn't get stucked in local minimuns, that means it is a good method to be used. The size of the initial population varied from 25 to 100 and the method was tested using a 120 problems database. The algorithm was implemented on MATLAB® where results show that it can consistently deliver a satisfactory good solutions for blocking flow shops problem.*

Key-words: *Genetic Algorithm, flow shop, block, makespan,scheduling.*

1. INTRODUCTION

As Technology is been developed, it is necessary to use tools and strategies to keep a company competitive at the market, differing products from others, chasing exclusivity, creating some valuable product characteristics for costumers, maintaining focus on the quality, brand image, technical assistance and a better support for costumers. Another strategy is to focus the products to a specific public, making a design for specific costumers. The last strategy is to reduce manufacturing costs, reducing the product price and increasing number of sold products. To Use the cost leadership strategy, it is necessary to automatize the manufacturing process, optimize design, set factory location near customers and suppliers and organize the production system. An effective and cheap method of increasing production using the same investment, minimalizing processing time and reducing costs is scheduling.

Scheduling is, according to (Pinedo, 1994), a form of decision-making that plays a crucial role in manufacturing and service Industries, that deals with allocation of resources to tasks over given time periods and its goal is to optimize on or more objectives. Using this tool is an excellent alternative to get to the objectives function of the paper`s problem, minimizing the conclusion time of the last job (Makespan) and the sum of conclusion time of all the jobs (flow time).

In this paper the problem is denoted as, according to (Pinedo, 2008), Fm / prmu, Sijk, block / Cmax, ∑Cj. Where, there are m machines and n jobs that must follow the same flow and setup depends on the manufacturing sequence. Also, blocking may occur in between two consecutive machines. In other words, become full and, if the next machine is occupied, the previous machine cannot complete the job. The objective in the problem is to minimize both the makespan and the flow time.

The problem was solved by using genetic algorithm (GA). According to (Mitchell, 1996), GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics−inspired operators of crossover, mutation, and inversion. Each chromosome consists of "genes" (e.g., bits), each gene being an instance of a particular "allele" (e.g., 0 or 1). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single−chromosome ("haploid") organisms; mutation randomly changes the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed.

The following section introduces genetic algorithms for the flow shop problems with blocking and sequence dependent setup times. 120 problems for 12 different combinations of number of jobs and machines, and the RDI method for the best PS found.

2. LITERATURE REVIEW

Choosing a method it is not a simple task, Chen et al. (1995) confirmed that GA based heuristic consistently gives better results than the SPIRIT and Ho and Chang's heuristic by comparing its results, after choosing it, it is necessary to study how block influences the environment, therefore Caraffa et al. (2001) observed genetic algorithm developed in their study delivered significantly better solutions for blocking flow shop involving up to 20 machines and 250 jobs as compared with a heuristic procedure developed in a recent Ph.D thesis in references of their paper.

Tseng and Lin (2010) made a hybrid genetic algorithm for no-wait flow shop scheduling problem, they noticed if you combine, for example, GA with the Insertion Search or Insertion Search with Cut-and-Repair, the makespan found compared to its mean is about 5% lower for the studied problem and Chaudhry and Khan (2012) minimized Makespan for the same no-wait environment as Tseng and Lin using genetic algorithm but introducing different ways for parents crossover, they divide the parents in 2 cuts separating the first 2 jobs and the last 3 from its core (this one will be the children's' core too), then starting from the second cut of the other parent you get all the jobs this one doesn't have and fill it.

Selecting an algorithms for initial solution is also an important task, Sanches et al. (2015) compared the use of different methods for obtaining an initial solution to the branch-and-bound algorithm in order to minimize the makespan in a flow shop environment with zero buffer, Takano and Nagano (2017) showed in a permutation flow shop problem with blocking and sequence and machine dependent setup times, four lower bounds for the Makespan were proposed and used in a branch-and-bound algorithm, the best lower bound was the LBTN2, a MILP model was also proposed for the problem and results showed the consistency of the proposed algorithm.

The innovation of this paper is to optimize two different objectives, Silva et al (2018) introduces an equation that makes possible analyzing more than one objective, establishing weights for each objective. Machine setups times were token from Takano and Nagano (2019) who created a database. The setup time value was uniformly distributed between 1 and 10, 50, 100 and 125 of the processing time value. In this article it was used database until 100.

3. METHODOLOGY

In this paper, it was used Genetic algorithm (GA) to minimize Makespan and flow time. The goal of this paper is to find the best size of the initial population (PS) that provides the best solutions for the multi-objective problem in good computational time.

As the GA proposed concentrate in a multi-objective problem, the genetic algorithm initially analyzes each single objective separately. GA_{best} and GA_{worst} are calculated for both objectives. The difference between GA_{best} and GA_{worst} is the way the parents are elected for the chromosome crossover. In GA_{best}, the chromosomes that obtained the best results are more likely to become parents; on the other hand, in GA_{worst}, the chromosomes that obtained the worst solutions are more likely to become parents. In the following paragraphs it will be explained how GA was construct.

For the initial population creation, candidate solutions were generated by different random permutations according to PS value, then theses candidates were evaluated according to its fitness for each objective, as explained before, but the main GA it uses a multi-objective criterion, as suggested by Silva et al. (2018), were its required pounds to analyze objectives, as equation (1) suggests, k refers to number of objectives, the sum goes from 1 to k, F is the value analyzed for objective k and Fk_best and Fk_Worst are the results obtained by implementing GA_{best} and GA_{worst} for this objective k:

$$F=\{\sum[Wk\{Fk(x)-Fk\_best(x)\}/(Fk\_worst(x)-Fk\_best(x))]\text{^}2\}\text{^}1/2 \quad (1)$$

Equation (2) Adapts it for this problem:

$$F=\{[Wmk\{Mk(x)-Mk\_best(x)\}/(Mk\_worst(x)-Mk\_best(x))]\text{^}2+[Wftk\{Ft(x)-Ft\_best(x)\}/(Ft\_worst(x)-Ft\_best(x))]\text{^}2\}\text{^}1/2 \quad (2)$$

Where Mk and Ft refer to makespan and flow time respectively and Wmk and Wft are their weights. In this paper, the value of the weights for both objectives were set to 0.5. According to the value of F, solutions were evaluated in a nondecreasing form to be possible to restrict possibility of parents (usually called parents in GA candidates that are used to create new possible different candidates, called child).

GA considers sequences of jobs as possible candidates, also called as chromosomes, a group of chromosomes is called population, to select members of this population to be parents and generate new children the top ε x 100% which have best fitness are possible to be selected, although parents may not be selected also, because there is a chance of them don't give birth to children, that is defined as probability of crossing two parents (PC).

For the parents crossing, each couple has the chance PC of generate two children, each children receive, job sequence of Parent 1 or Parent 2, if the number of jobs *n* is smaller than 5, only the first and the last job of children are

deleted, otherwise the first and the last two jobs are deleted, then jobs attributed to this child are compared to the other parents jobs, from left to right the non-equal ones are filled in each child.

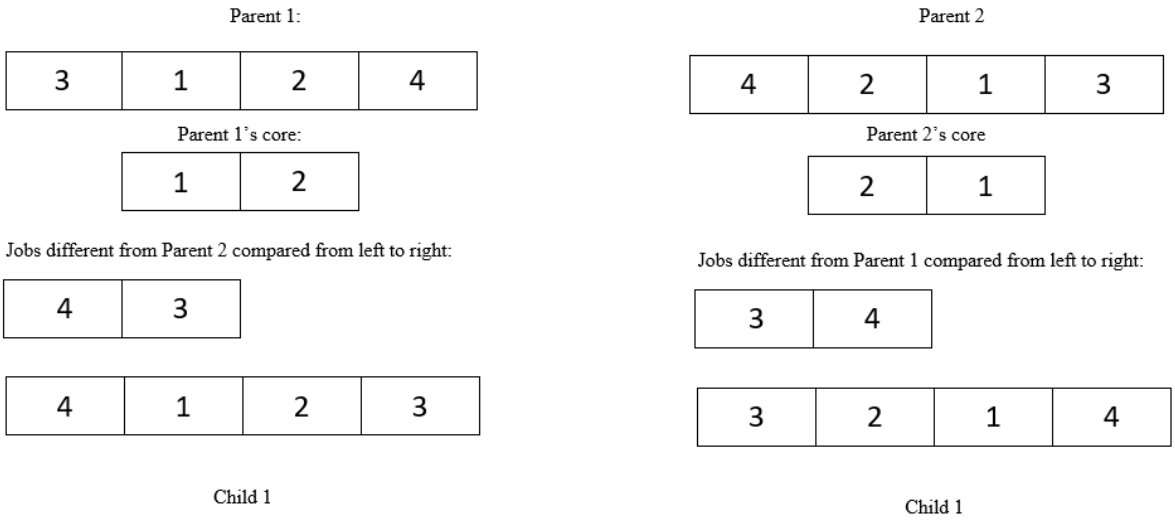

Figure 1: Parents crossing

Now, with two new population members, there is a chance PM of mutating, where 2 jobs are switched, so according to mutation percentage, two values are possibly generated from 1 to n, these positions are spotted and switched as shown in Fig 2:

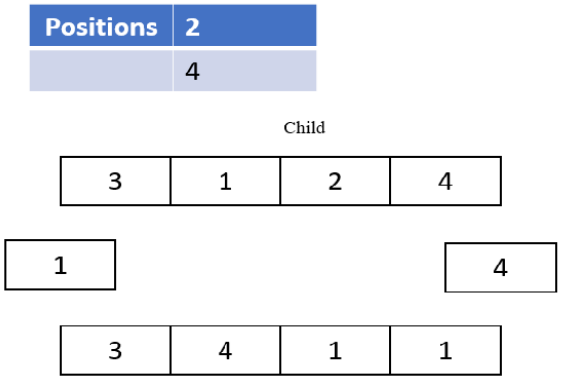

Figure 2: Permutation

This crossing and mutating procedure will repeat until population size double, selecting random parents of the top fraction ε each iteration, when the process get to its end, that means Ps initial is doubled, theses solutions are evaluated again according to multi-objective F value, and the last half of population which contains the worst candidates are killed and it means that this generation is complete. This procedure will repeat until the counter Gen get pass through all generations.

To use GA, first it is necessary to set ε, PS, PC, Pm, and the number of generations n_{gen}. All the parameters for the GA, except for the PS, were taken from Caraffa et al. (2001), where they found out good values for these variables. The parameters used were: PC=0.9, Pm=0.05, n_{gen}=25 and ε=0.35. As the influence of the PS to result is the objective in this paper, the algorithm was tested with PS=25;50;75;100.

Tests were performed using the problem database provided by Taillard (1993), which is composed of 120 instances ranging from 20 jobs and five machines to 500 jobs and 20 machines. The setup times used in this paper was proposed by Takano and Nagano (2017). Both databases generated the processing times and setup times by uniformly distributing

values between 1 and 99. Tests were performed in a Dell Inspiron Gaming Edition i15-7567-A20P, windows 10, Intel® Core™ i7-7700HQ CPU @ 2.80GHz, 8 GB RAM.

The values of makespan, flow time and computational time (CPU Time) found are recorded, and in the end all these values are used to measure the performance of each PS chosen by the relative deviation index (RDI) Method.

4. RESULTS

In table 1, all classes of problems from Taillard (1993) database were considered. For each PS and each class of problems, it was calculated the mean CPU time and the mean value of the objective function (F) of all 10 problems, and the results are shown in table 1. Also, in table 1 it is presented the mean values of the mean CPU time and the mean value of the objective function (F) obtained in all 12 classes of problems.

Table 1. Results of Objective Function (F) and its computational time for each problem size.

PS	25		50		75		100	
Problem	**CPU Time (s)**	**F (%)**	**CPU Time (s)**	**F (%)**	**CPU Time (s)**	**F (%)**	**CPU Time (s)**	**F (%)**
Pjk 20 x 5	0.34013	0.63105	0.54080	0.41544	0.76526	0.23176	1.04343	0.09051
Pjk 20 x 10	0.54925	0.62338	1.01159	0.58093	1.55977	0.49899	2.06823	0.70168
Pjk 20 x 20	1.40073	0.70317	2.78909	0.55685	4.22846	0.48906	5.69812	0.64477
Pjk 50 x 5	0.64711	0.58141	1.26942	0.61715	1.87397	0.55916	2.54055	0.54785
Pjk 50 x 10	1.33229	0.63435	2.57582	0.60089	3.94863	0.50531	5.28154	0.55507
Pjk 50 x 20	3.61621	0.65788	7.25188	0.53385	10.88239	0.56862	14.59101	0.55006
Pjk 100 x 5	1.34719	0.62418	2.53085	0.58880	3.90372	0.56533	5.19615	1.21362
Pjk 100 x 10	2.66759	0.65540	5.35126	0.59970	8.05468	0.53661	10.77732	0.89219
Pjk 100 x 20	7.48490	0.66340	14.82436	0.54334	22.23235	0.56154	29.66855	1.18698
Pjk 200 x 10	6.66289	0.61923	11.15048	0.59879	16.83543	0.53268	22.45143	0.58475
Pjk 200 x 20	15.87072	0.65478	30.70808	0.56461	46.19583	0.52657	61.51982	0.58523
Pjk 500 x 20	40.78393	0.64503	81.60871	0.56107	124.82631	0.52531	163.39454	0.64900
Mean	**6.89191**	**0.64111**	**13.46770**	**0.56345**	**20.44223**	**0.50841**	**27.01922**	**0.68348**

Figure 3 represents how the objective values variate according to the size of the problem, considering all the 120 problems divided in 12 calsses as shown in Problem columnn of Table 1. Figure 4 represents how computational time variates according to the problem size. Analyzing Table 1 and Figure 3 it is possible to see that even being the option that gives more initial solutions PS=100 has the largest value of F and the highest computational time, therefore being the worst choice. PS=75 is the population size that got the smallest value of F. It is also possible to confirm that computational time is directly proportional to population size as it is shown in Table 1 and verified in Figure 4.

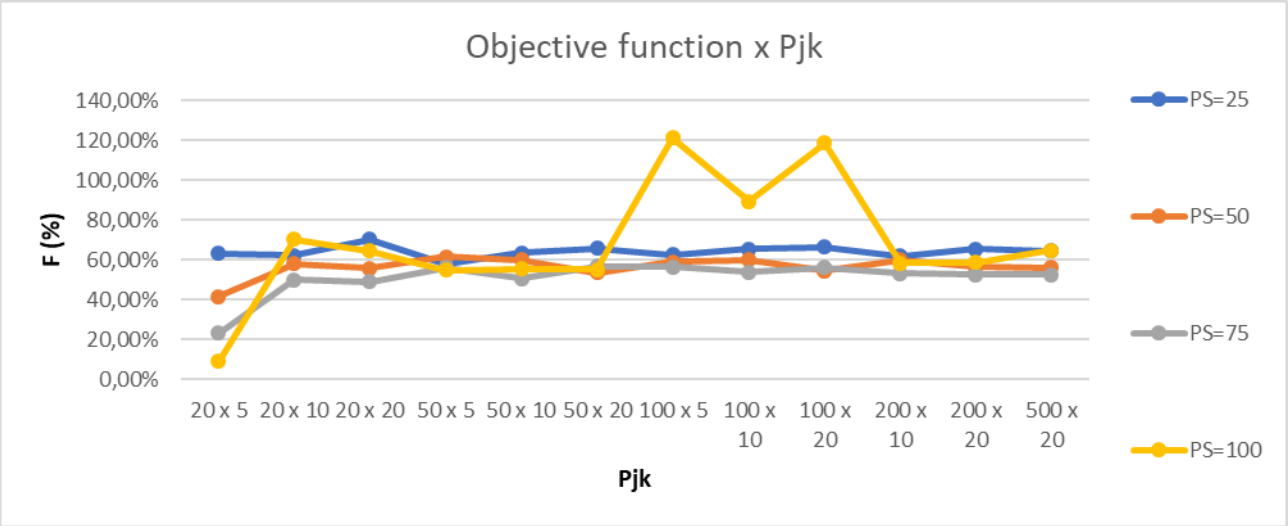

Figure 3: Objective function F

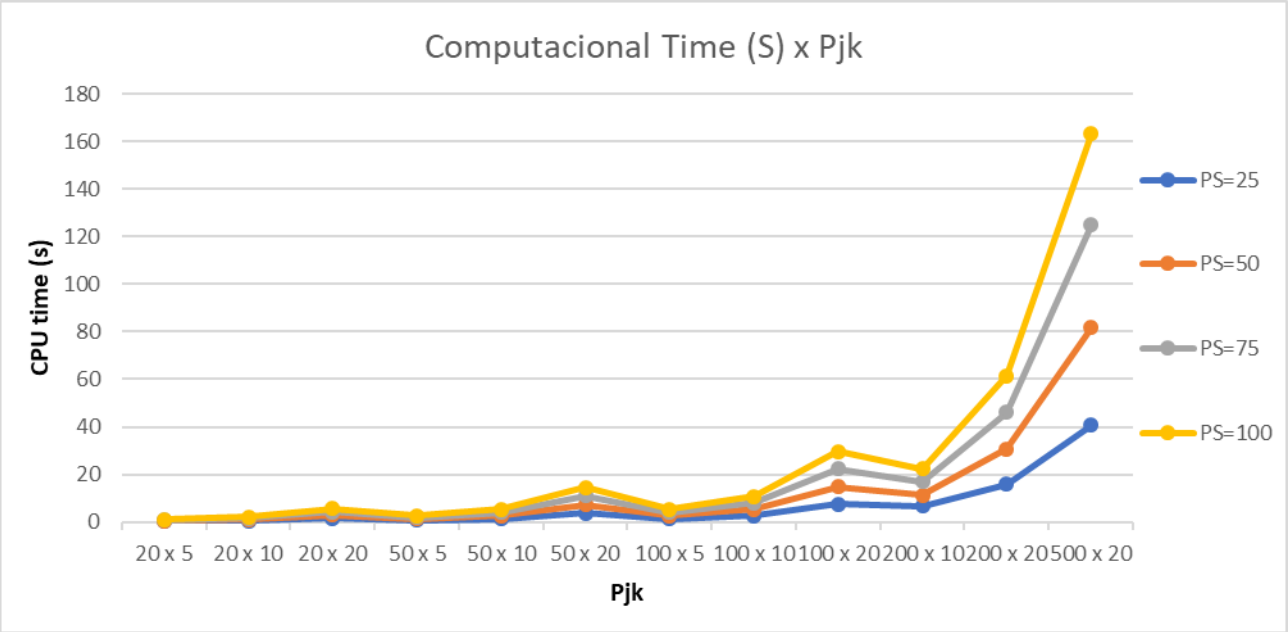

Figure 4:Computational time

In Figure 5, equation 1 was applied to minimize time and F value because PS=75 has the best F but PS=25 has the best computational time, for this the best and worst CPU mean time and F mean value were taken from table 1, both weights of equations were set in 0.5.

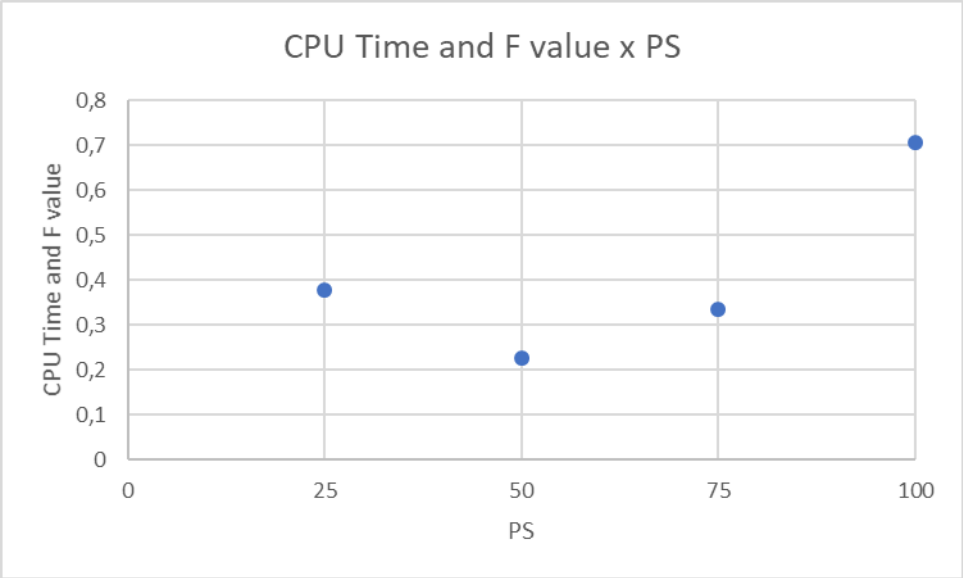

Figure 5: CPU time and F value according PS

Therefore, although PS=50 does not have the best F nor the best CPU time, it is considered the best size for the initial population among the ones tested. The loss in the percentage of F is compensated by the gain in CPU Time.

5. CONCLUSION

GA showed itself a good method for solving flow shop problems because it is efficient and fast. In order to keep a low computational time, it is necessary to reduce PS as possible maintaining good values for F, as shown before, CPU time is directly proportional to PS value.

Equation 1 was utilized because Makespan and flow time had different shapes, since there is this difference it isn't possible to analyze them comparing their values directly, thus equation 1 makes the comparison possible by normalizing their values, as both weights were set in 0.5, both objectives have the same influence in F value.

For future papers, it is intended to test how the algorithm responds as the setup values varies. Tests can be performed by varying the distribution of the setup times, for example: from 1 to 25; from 1 to 50; from 1 to 120; etc. Also, tests can be performed with a higher range of population sizes, for a better analysis of the influence of the PS. Finally, the GA method can be compared to other heuristics methods, such as Iterated Greedy.

6. REFERENCES

Caraffa, V., Ianes, S., Bagchi, T.P. and Sriskandarajah, C., 2001. "Minimizing makespan in a blocking flowshop using genetic algorithms". *International Journal of Production Economics*, Vol. 70, pp. 101–115.

Chaudhry, I. A. and Khan, A. M., 2012. "Minimizing makespan for a no-wait flowshop using genetic algorithm". *Indian Academy of Sciences*, Vol. 134, pp. 695–707.

Chen, C., Vempati, V.S., Aljaber N., 1995. "An application of genetic algorithms for flowshop problems". *European Journal of Operational Research*, Vol. 80, pp. 389–396.

Melanine, M., 2008. *An Introduction to Genetic Algorithms*. A Bradford Book The MIT Press, Massachusetts, 1nd edition.

Sanches, F.B., Takano, M.I. and Nagano, M.S., 2015. "A hybrid genetic algorithm for no-wait flowshop scheduling problem". In *Proceedings of the 23nd International Congress of Mechanical Engineering - COBEM 2015*. Rio de Janeiro, Brazil.

Silva, C.A.X., Taketa, E., Koroishi, E.H., Lara-Molina, F.A. and Faria, A.W., 2018. "Optimization of parameters of a modal active control in a beam of composite material". *MATEC Web of Conferences*, Vol. 211, 19002.

Takano, M.I. and Nagano, M.S., 2017. "A branch-and-bound method to minimize the makespan in a permutation flowshop with blocking and setup times". *Cogent Engineering*, Vol. 10, pp. 37–50.

Takano, M.I. and Nagano, M.S., 2019. "Evaluating the performance of constructive heuristics for the blocking flowshop scheduling problem with setup times". *International Journal of Industrial Engineering Computations*, Vol. 10, pp. 37–50.

Tseng, L. and Lin, Y., 2010. "Minimizing makespan in a blocking flowshop using genetic algorithms". *International Journal of Production Economics*, Vol. 128, pp. 144–152.

ZINI, E.O.C., 2009. *Algoritmo Genético Especializado na Resolução de Problemas com Variáveis Contínuas e Altamente Restritos*. Master's dissertation, Universidade Estadual Paulista "Julio de Mesquita Filho", Ilha Solteira, Brasil.

7. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.