Transgenic Genetic Algorithm to Minimize the Makespan in the Job Shop Scheduling Problem

Abstract

In recent years, several research studies have been conducted that use metaheuristics to calculate approximations of solutions for solving NP-Hard problems, within this class of problems there is the Job Shop Scheduling Problem (JSSP), which is discussed in this study. Improved solutions to problems of this type have been created for metaheuristics in the form of additional operators. For the Genetic Algorithm (GA) the transgenic operator has recently been created, whose operation is based on the idea of "genetically modified organisms", with the proposal to direct some population of individuals to a more favorable solution to the problem without removing the diversity of the population with a lower cost of time. In this study, our main contribution is an adaptation of the GA with transgenic operator to the JSSP. The results obtained by the proposed method were compared with three papers in the literature: one using GA, one using Adaptive GA and another using Ant Colony Optimization. The results confirm that the GA used with the transgenic operator obtains better results in a shorter processing time in comparison to the other techniques, due to its better targeting in the search space.

Introduction

The job shop scheduling problem (JSSP) is a combinatorial optimization problem defined in the literature as in the NP-Hard class (Lu, Shi et al. 2018). Therefore, it is recommended the use of heuristic, metaheuristic and stochastic algorithms to optimize NP Hard class problems (Hasan, Sarker, and Essam 2010).

The JSSP is part of a class of problems among the job-based scheduling problems. This class represents a research area of great importance in current studies, such as flexible job scheduling problems (FJSPs), parallel machine scheduling problems (PMSPs), test task scheduling problems (TTSPs) and others (Lu, Shi et al. 2018). Specifically, in this work, we approach the class of combinatorial optimization problems known as JSSP. In the following paragraphs, are presented the current studies in the literature that approach some type of job-based scheduling problems.

Nguyen, Zhang, and Tan (2018) proposed a study of the dynamic flexible job shop scheduling problem with a new genetic programming algorithm (GP), entitled adaptive charting GP (ACGP), the proposed algorithm. The ACGP can balance its exploration, getting exploitation better than the existing surrogate-assisted algorithm. The proposal performed better than standard genetic programming algorithm.

Romero et al. (2018) proposed a study of the flexible job shop scheduling problem (FJSSP) with Lot Streaming with the Tabu Search (TS) algorithm, the study was compared with a mathematical programming solver, GUROBI. The algorithm obtained better results surpassing the upper limits found of GUROBI.

Araujo, Arroyo, and Tavares (2018) proposed a study of the flow shop scheduling problem (FSSP) with a hybrid meta-heuristic, a Variable Neighborhood Search (VNS) heuristic coupled with Iterated Greedy (IG) algorithm, named VNS-IG. The objective of the problem was to minimize the maximum completion time (makespan). The VNS-IG obtained better results than the SS algorithm.

Öztop et al. (2018) proposed a study of the hybrid flow-shop scheduling problem (HFSP) using the Iterated greedy algorithms, IG and IGALL. The objective variable was to minimize total flow time and has been tested in HFSP instances from the literature. The authors emphasize that one of the main contributions of the study was that the results of flow time criterion have been reported for the HFSP benchmark suite for the first time.

Dao et al. (2018) proposed a study of the JSSP with an algorithm based on parallel versions of the bat algorithm (BA), using as objective variable makespan. The algorithm presented better convergence and competitive results than BA traditional and particle swarm optimization (PSO).

Morandin, Kato et al. (2008) proposed a study to solve the Production Scheduling of Manufacturing Systems, the study has been tested in a benchmark of the type JSSP with a traditional genetic algorithm (GA) as a search method, using as objective variable the makespan measure. The GA achieved competitive results in a shorter processing time.

Morandin, Sanches et al. (2008) added an improvement in the GA proposed in Morandin, Kato et al. (2008), adaptive rules were included to the algorithm, entitled adaptive genetic algorithm (AGA). The crossover and mutation rate are dynamically adjusted according to the individual's

fitness value. The study has been tested in a benchmark of the type JSSP. The AGA presented solutions with response time acceptable.

Kato, Morandin and Fonseca (2010) proposed a study to solve the Production Scheduling of Manufacturing Systems. The study has been tested in a benchmark proposed in Morandin, Sanches et al. (2008). The authors use an Max-Min Ant System algorithm as a search method. The proposal was compared with Morandin, Sanches et al. (2008) and obtained quality solutions in a shorter processing time.

In the literature it is possible to find several studies belonging to the job-based scheduling problems class that approach the problem with AG, some studies as: (Peng et al. 2018), (Lu, Wu et al. 2018), (Hosseinabadi et al. 2019), (Kundakci and Kulak 2016), (Kurdi 2016), (Asadzadeh 2015) and (Driss, Mouss, and Laggoun 2015).

The GA is a metaheuristic widely used in current studies due to several advantages that this algorithm has, but it also has some disadvantages such as not solve complex problems easily (Guo, Wang, and Han 2010). The GA has as one of its main disadvantages the high consumption of resources, that is, domain of large solutions will use longer search time (Kazemi, Mohamed, Shareef, and Zayandehroodi 2012; Nie, Gao, Li, and Li 2012).

Amaral and Hruschka (2014) have developed an operator for evolutionary algorithms entitled Transgenic Operator. This operator was inspired by genetic engineering, in which there is the possibility of manipulating the genetic material of individuals by adding features that are believed to be important. This type of approach can be understood as a strategy of elitism focused on specific genes. The Transgenic Operator must direct a portion of individuals of the population for a better solution, without loss of diversity of the population and in a smaller cost of time.

The objective of this work is the application of an alternative version of Transgenic Operator (Amaral and Hruschka 2014) in the job shop scheduling problem.

In this paper, we approached the reasoning proposal used in (Kato, Morandin, and Fonseca 2010; Morandin, Kato et al. 2008; Morandin, Sanches et al. 2008), which use the same benchmark for the job shop problem and are solved, respectively, by the metaheuristics GA, AGA and Ant Colony Optimization (ACO). In this way, the comparison of the results of our method with the methods of such studies becomes more natural, since we will use the same benchmark.

The remainder of this paper is organized as follows. Section 2 contains the JSSP specification and the fundamentals of GA and description of the Transgenic Operator. The components of the proposed algorithm are presented in Section 3. The computational experiments and analyses of the obtained results are presented in Section 4. Finally, Section 5 presents the conclusions of the paper.

Problem Description

Job Shop Scheduling Problem

In this context, in a manufacturing system, there is a set of n products (jobs) $\{P_1, P_2, \ldots, P_n\}$ that are produced by manufacturing, and such products make shared use of a set of m machines $\{M_1, M_2, \ldots, M_m\}$. A job $\{P_1, P_2, \ldots, P_n\}$ contains a set of operations and a predetermined sequence of machines. Each operation makes use of one of the machines $\{M_1, M_2, \ldots, M_m\}$ for a predetermined time interval to complete a job. A schedule can be defined as the assignment of operations at predetermined time intervals on a machine (Dao et al. 2018).

Search with Genetic Algorithm

Genetic Algorithms (GAs) were developed in the 1970s by Holland (Holland 1975) with the objective of optimizing complex and non-linear systems. This type of technique has a strong appeal to biological inspiration derived from the theory of evolution to perform its operation, so that its use does not require very elaborate mathematical theories.

Many improvements have been implemented over the last few years (Antonio and Coello 2017) to the GAs that Holland formulated in his initial work (Holland 1975) and presented in the classic 1992 book (Holland 1992). However, all the improved GAs present in the current specialized literature maintains as the main sequence of steps the one originally presented in (Holland 1992) Algorithm 1.

Algorithm 1: A Genetic Algorithm Pseudocode.

(1) Generate initial population $\Omega_0 = \{C_1, C_2, \ldots, C_n\}$
(2) Evaluate the fitness of the initial population
(3) **Repeat**
(4) Select individuals for crossover
(5) Apply crossover operator
(6) Apply mutation operator
(7) Evaluate new individuals
(8) Generate a new population: Ω_{it+1}
(9) **Until** Termination criterion is satisfied

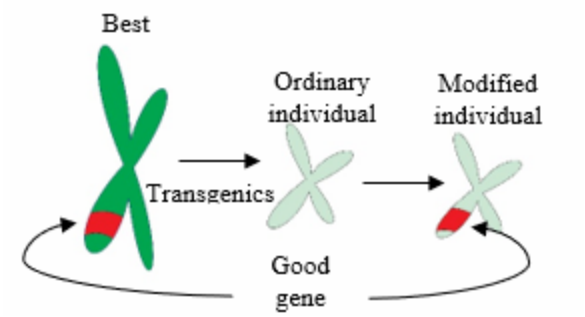
The generation of a new population, as done in step (8) of Algorithm 1, generally takes into account some own insertion technique so that only the n best individuals from the iteration are maintained and do not change the size of the population.

Transgenic Operator

In order to simulate the biological advances of genetic engineering, Amaral and Hruschka (2014) proposed the use of transgenic technique in GA. The concept of transgenics is to transfer, from one generation to another, genes

that probably describe a good feature, as exemplified in the Figure 1.

Figure 1. Transgenic Process.



For example, vitamin supplementation of maize is used in developing countries to avoid that the population, usually with food habits based mainly on cereals, suffers from lack of vitamins. This supplementation can be done with the use of transgenic planting (Naqvi et al. 2009).

The addition of this concept to GA occurs in the form of an operator, as represented in Algorithm 2.

Algorithm 2. A Transgenic Genetic Algorithm Pseudo-Code.

(1)	Generate initial population $\boldsymbol{\Omega_0} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, \dots, \boldsymbol{C_n}\}$
(2)	Evaluate the fitness of the initial population
(3)	Repeat
(4)	Select individuals for crossover
(5)	Apply crossover operator
(6)	Apply mutation operator
(7)	Evaluate new individuals
(8)	Generate a new population: $\boldsymbol{\Omega}_{it+\frac{1}{2}}$
(9)	Apply **transgenic** operator
(10)	Evaluate modified individuals
(11)	Generate modified population: $\boldsymbol{\Omega}_{it+1}$
(12)	Until

In **Erro! Fonte de referência não encontrada.**, two inserts of individuals are carried out: one in the step (8) and another in the step (11). However, only the population Ω_{it+1} is maintained in the process, since the population $\Omega_{it+\frac{1}{2}}$ is an intermediate population, from which the transgenic individuals are made in step (9). In this way, the population Ω_{it+1} is formed by the individuals of $\Omega_{it+\frac{1}{2}}$ together with the transgenic individuals.

The transgenic operator codification is described in detail in the next section.

The Proposed Genetic Algorithm Model for JSSP

Chromosome Codification

It is assumed that the JSSP treated in this work has m machines to process n products $\{P_1, P_2, \dots, P_n\}$. That is, it is a JSSP $n \times m$. In addition, each product P_i presents a set of scripts $\{R_{i,1}, R_{i,2}, \dots, R_{i,n_i}\}$, which define their processing order in machines.

Thus, the chromosome (C) of the proposed method is formulated according to Equation (1) below:

$$\mathrm{C} = \left(\left[P_{i_1}, R_{i_1,j_1}\right], \left[P_{i_2}, R_{i_2,j_2}\right], \dots, \left[P_{i_n}, R_{i_n,j_n}\right]\right), \quad (1)$$

in which, $i_k \in \{1,2, \dots, n\}$ and $j_k \in \{1,2, \dots, n_{i_k}\}, \forall k$.

Thus, the problem chromosome is formed by the genes $g_i = \left[P_i, R_{i,j(i)}\right]$ and hence C represents a product processing order in JSSP.

Fitness Function

The objective function of this work is the time taken to process the products of the JSSP $n \times m$ according to the configuration given in the input chromosome. Thus, the definition of this function is given in Equation ((2):

$$\mathrm{fit(C)} = \mathrm{fit}\left(\left[P_{i_1}, R_{i_1,j_1}\right], \left[P_{i_2}, R_{i_2,j_2}\right], \dots, \left[P_{i_n}, R_{i_n,j_n}\right]\right) = Mks, \quad \textbf{(2)}$$

in which, Mks is the makespan value of configuration C. More information about the makespan measure can be found in (Morandin, Kato et al. 2008).

Transgenic Operator Codification

In this work, we use a more simplified version of the original transgenic operator (Amaral and Hruschka 2014), since we do not define a priori which genes are more relevant in the chromosomal modelling of the resolution of a JSSP. In this way, the transgenic operator proposed in this work does not have a ranking stage of relevance for the genes, which does not affect its performance, as presented in Section 4.

We propose, in a preliminary way, that the genes set for use in the transgenic operator are the index genes $J = \{j_1, j_2, \dots, j_{N_{\mathrm{Trans}}}\}$, so that the elements of J are N_{Trans} different numbers randomly taken on the set $\{1,2,\dots,n\}$, where n is the number of genes in a chromosome and N_{Trans} is the number of genes to be transferred in the operator.

In order to control the reduction of population diversity, we propose the use of $N_{\mathrm{Trans}} \leq \sqrt{n}$[1], since if genes are replicated in large quantities, transgenic individuals may present endogenous phenomena.

In order for the concept of transgenesis to be maintained, we propose to transfer genes from a model individual, which is the individual with the best fitness, to the worst

[1] In this work, we take $N_{\mathrm{Trans}} = \sqrt{n}$.

individuals. That is, in each generation t, we take the best individual c^* and transfer its genes, whose indices belong to J, to the n_{Trans} worst individuals of the same generation.

Thus, the transgenic operator is modelled according to the function $\text{trans}_{c^*}(\cdot)$ presented in Equation (3):

$$\begin{aligned}\text{trans}_{c^*}(C) &= \text{trans}_{c^*}\left(\left[P_{i_1}, R_{i_1,j_1}\right], \dots, \left[P_{i_n}, R_{i_n,j_n}\right]\right) \\ &= \text{trans}_{c^*}\left(g_{i_1}, \dots, g_{i_n}\right) \qquad \textbf{(3)} \\ &= \left(g_{i_1}, \dots, g^*_{j_1}, \dots, g^*_{j_2}, \dots, g^*_{j_{N_{\text{Trans}}}}, \dots, g_{i_n}\right),\end{aligned}$$

in which, $c^* = (g^*_{k_1}, g^*_{k_2}, \dots, g^*_{k_n})$ is the best individual in the current generation.

In this way, the transgenic operator is defined in Algorithm 3.

Algorithm 2: Transgenic operator

Input:	n_{Trans}	Number of individuals to get transgenics
	Ω_t	Current population
	J	Index of genes to be transferred
(1)	$c^* \coloneqq$ Individual with best fitness on Ω_t	
(2)	$\{w_1, w_2, \dots, w_{n_{\text{Trans}}}\} \coloneqq$ The n_{Trans} worst individuals in Ω_t	
(3)	For $i = 1$ to n_{Trans} do:	
(4)	$w_i^* \coloneqq \text{trans}_{c^*}(w_i)$	
(5)	End	
Output:	$\{w_1, w_2, \dots, w_{n_{\text{Trans}}}\}$	Transgenic individuals

Basic Operators

The genetic algorithm we developed is based on the genetic algorithm of (Morandin, Kato et al. 2008) used in the resolution of JSSPs of size 9 x 9. In this way, all the standard operators of our GA, such as crossover and mutation, are the same operators described in (Morandin, Kato et al. 2008). For the selection operator, we use the technique of roulette wheel and for insertion operator we use elitism.

Results and Discussion

Experimental Settings

The method described in this paper was evaluated in a specific job shop scheduling problem of size 9 × 9, which consists of a problem of n=9 products (jobs) and m=9 machines, which was found and detailed in [8–10, 26](Kato, Morandin, and Fonseca 2009; Kato, Morandin, and Fonseca 2010; Morandin, Kato et al. 2008; Morandin, Sanches et al. 2008). The number of evaluations was set to 35 for each technique in order to use the non-parametric Wilcoxon tests (Veldhuizen and Lamont 2000) to determine if our method presents competitive results to the compared metaheuristics. The algorithms tested were coded in Matlab software. All tests were run on a notebook with i7 processor and 16GB RAM.

Results of the Proposed Algorithm and Comparisons with Other Works

In order to compare the efficiency of the proposed method (GA-Trans), we implemented some metaheuristics already successfully used in JSSP: GA (Morandin, Kato et al. 2008); Adaptive GA (AGA) (Morandin, Sanches et al. 2008); and Ant Colony Optimization (ACO) (Kato, Morandin, and Fonseca 2010).

We try to follow as closely as possible the settings presented in each work for honest results. However, the configurations of our method are more similar to GA and AGA metaheuristics configurations, which makes sense, since these algorithms differ only in the use of specific operators. Thus, the configurations of the GA, AGA and GA-Trans techniques are presented in Table 1 and the configurations used by the ACO are set out in Table 2.

Table 1: State-of-the-art configuration on GAs

	GA	AGA	GA-Trans
Number of chromosomes	30	30	30
Crossover Rate	0.8	0.8	0.8
Mutation Rate	0.05	0.05	0.05
Iterations	200	200	200
Transgenic rate	-	-	0.4
Stopping Criterion	Iterations	Iterations	Iterations

Table 2: State-of-the-art configuration on ACO

	ACO
Number of Ants	50
$\boldsymbol{\alpha}$	1
$\boldsymbol{\beta}$	2
$\boldsymbol{T_{max}}$	10
$\boldsymbol{T_{min}}$	0.25
Evaporation	0.02
Iterations	75
Stopping Criterion	Iterations

The 35 tests are show in Table 3. So, the maximum value obtained by each technique is colored red and the minimum value is colored blue.

Table 3: Results of **35** Tests

	ACO	GA	AGA	GA-Trans
1	**4632**	4698	**5015**	4670
2	4669	4936	4677	4944
3	**4977**	4752	4673	**4632**
4	4945	4956	4640	4635
5	4929	4694	4741	4691
6	4872	4996	4718	4929
7	4746	4917	4944	4848
8	4754	4981	4688	**4632**

9	4693	4954	4989	4688
10	4901	5051	**4632**	4848
11	4968	4848	4659	4635
12	4736	4991	4934	5019
13	4688	4705	4921	4688
14	4956	4718	4956	4635
15	4895	4988	4925	4929
16	4734	4725	4945	4635
17	4788	5042	4880	4860
18	4782	4945	4848	**4632**
19	4704	4934	4956	4688
20	4899	5068	4945	4677
21	4752	4891	4848	**5051**
22	4785	**4669**	4951	4670
23	4752	4951	4919	4776
24	4929	4958	5006	4656
25	4860	4848	4968	4691
26	4763	4710	4693	4718
27	4688	4925	4705	4670
28	4642	4898	4984	4979
29	4688	4848	4759	4635
30	4909	4945	4944	4635
31	4693	4848	4735	4670
32	4692	4679	4635	4654
33	4898	**5168**	4929	4929
34	4646	4705	4956	4635
35	4946	4901	**4632**	4677

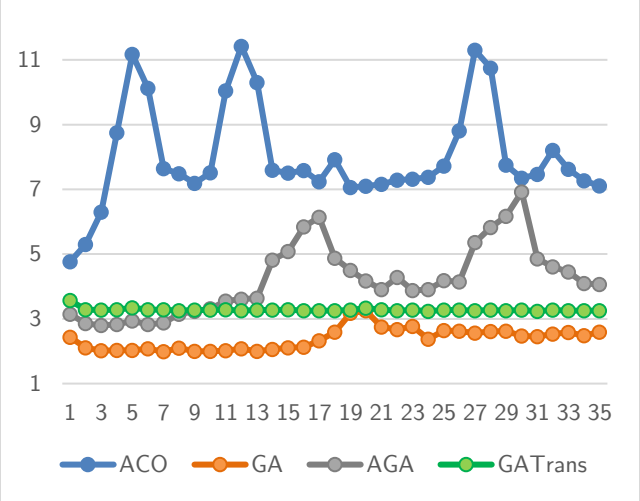
Analyzing Table 3 and Table 4, it can be concluded that the proposed technique presented, on average, makespan values that are smaller than the other techniques. In addition, GA-Trans presented the smallest makespan (4632) of all of the techniques discussed when considering the 35 evaluations conducted in each. In addition, GA-Trans presents as the most often occurring value the value 4635, which is a makespan value that is less than the minimum value presented by the GA and a value that is very close to the minimum makespan presented by the methods.

Table 4: Statistical Measures.

	ACO	GA	AGA	GA-Trans
Mean	4797.45	**4881.22**	4838.57	**4741.74**
Standard Deviation	**110.37**	130.17	**134.31**	131.39
Minimum Value	**4632**	**4669**	**4632**	**4632**
Maximum Value	**4977**	**5168**	5015	5051
Mode	4688	4848	**4956**	**4635**
Average of time taken (s)	**8.01**	**2.38**	4.20	3.27

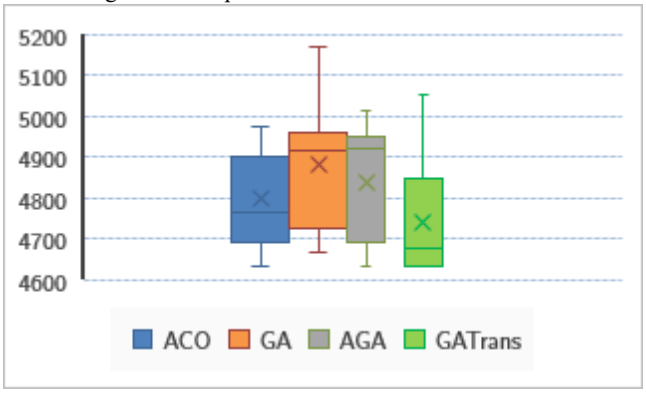
With respect to the average time of the GA-Trans, it can be affirmed that it is very competitive to GA-like methods and usually takes only 40% of the time spent by the ACO, as can be observed in more detail in Figure 2.

Figure 2: Time taken in **35** tests for each technique.



Although GA-Trans does not present the smallest worst makespan, it can be observed in the box plot of Figure 3 that it is a discrepant value of the technique. In fact, we can see that GA-Trans is the method that usually presents the best results in comparison with the other techniques.

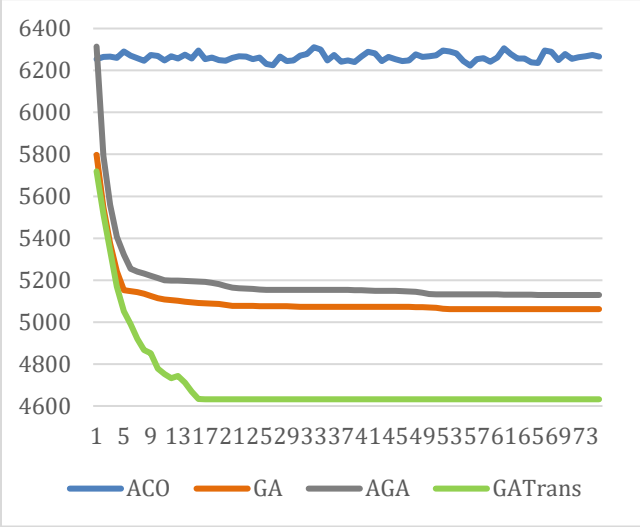
Figure 3: Box plot of the methods' results.



Convergence graphs were constructed of the proposed algorithm and the three algorithms tested, to evaluate the evolution of the entire population of the algorithm when iterations of the method are advanced. Specifically, all methods were used to generate a solution, and all individuals involved in the process were evaluated. In this way, graphically, the *y*-axis shows the average of all of the ma-

kespan values reached by the individuals (chromosomes or ants) of a population along the iterations of each technique, and the x-axis shows its number of iterations (generation). The purpose of these graphs is to demonstrate how fast or slow the algorithm is in finding an optimal solution. As seen in the convergence graphs of the algorithm, the GA with the transgenic operator shown in Figure 4, which is being directed through the application of transgenics of the most significant genes, has a convergence that requires fewer iterations if compared to a simple GA, to an Adaptive GA or to an ACO. With this finding, we note the advantage in a faster convergence that a GA with a transgenic operator can offer. Furthermore, according to the graph, there is no consensus among ACO ants in the observed evaluation, since the ants that find the minimum makespan do not significantly change the mean of the whole population.

Figure 4: Mean fitness function of a chromosome or ant population over **75** iterations of each method.



The Wilcoxon test is used to infer if two samples come from the same distribution and, if they are not, the test also classifies which sample is composed of statistically lower values. Thus, we use this statistical test to decide whether GA-Trans is statistically equivalent to some other metaheuristic evaluated. As observed in Table 5, assuming as initial hypothesis (H_0) that GA-Trans is equal to some other metaheuristic, the Wilcoxon test presents zero or almost zero p-values (0.0026 and 0.0072), which means that the initial hypothesis must be discarded as it is very unlikely. Similarly, assuming that GA-Trans has makespan values statistically lower than the values presented by other metaheuristics, the Wilcoxon test presents p-values equal to or very close to 1, which guarantees that GA-Trans presents statistically lower values.

Table 5: Wilcoxon test

	H_0	p-Value	Confidence Level
GA-Trans ×	GA-Trans=GA	0	95%
GA	GA-Trans<GA	1	95%
GA-Trans ×	GA-Trans=GA	0.0026	95%
AGA	GA-Trans<GA	0.9987	95%
GA-Trans ×	GA-Trans=GA	0.0072	95%
ACO	GA-Trans<GA	0.9965	95%

Conclusion

The objective of the paper was to develop an alternative version of the transgenic operator proposed by Amaral and Hruschka (2014) to reduce the makespan in job shop scheduling problem. The proposal was evaluated, and the results obtained were compared to other approaches proposed in related work (Kato, Morandin, and Fonseca 2010; Morandin, Sanches et al. 2008; Morandin, Kato et al. 2008), using as an evaluation criteria the minimization of the makespan value and the time to obtain the response.

Comparing the makespan values obtained, there was a tendency of improvement of the proposed algorithm in 82.86% of the cases in comparison to the results obtained with the Genetic Algorithm and in 71.43% with the Adaptive Genetic Algorithm. In comparison with the ACO technique, there was a tendency to improve the proposed algorithm in 65.71% of the cases. The mean execution time of the proposed algorithm was 3.27 seconds, while the mean time spent by the GA was 2.38 seconds; the Adaptive GA used 4.20 seconds, and the ACO used 8.01 seconds, i.e., there was an increase of 37.39% when the proposed technique was compared with GA, and there was a reduction in the processing time of the proposed algorithm of 22.14% with respect to the AGA and a reduction of 60% with respect to the ACO. In addition, by comparing the values

of the makespan obtained for the problem addressed, it is possible to conclude by means of the Wilcoxon statistical test, with 95% confidence, that the proposed method will have better results than the results obtained by the GA, Adaptive GA and ACO.

The genetic algorithm with a transgenic operator is promising in solving the JSSP. Thus, it is convenient that in future studies, the proposed algorithm is applied in problems similar to the JSSP, since the GA with transgenic operator obtained more significant results when compared to other metaheuristics. In this way, it is possible to work equivalently when applied to other combinatorial problems. It would also be interesting to study possible techniques to determine the most significant genes that are passed in the transgenic stage.

References

AMARAL, Laurence Rodrigues; HRUSCHKA JR, Estevam Rafael. Transgenic: An evolutionary algorithm operator. **Neurocomputing**, v. 127, p. 104–113, 2014.

ANTONIO, Luis Miguel; COELLO, Carlos A Coello. Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art. **IEEE Transactions on Evolutionary Computation**, v. 22, n. 6, p. 851–865, 2017.

ARAUJO, Matheus de Freitas; ARROYO, José Elias C; TAVARES, Ricardo G. An effective hybrid meta-heuristic for a heterogeneous flow shop scheduling problem. 2018, [S.l: s.n.], 2018. p. 245–252.

ASADZADEH, Leila. A local search genetic algorithm for the job shop scheduling problem with intelligent agents. **Computers & Industrial Engineering**, v. 85, p. 376–383, 2015.

DAO, Thi-Kien et al. Parallel bat algorithm for optimizing makespan in job shop scheduling problems. **Journal of Intelligent Manufacturing**, v. 29, n. 2, p. 451–462, 2018.

DRISS, Imen; MOUSS, Kinza Nadia; LAGGOUN, Assia. A new genetic algorithm for flexible job-shop scheduling problems. **Journal of Mechanical Science and Technology**, v. 29, n. 3, p. 1273–1281, 2015.

GUO, Pengfei; WANG, Xuezhi; HAN, Yingshi. The enhanced genetic algorithms for the optimization design. 2010, [S.l: s.n.], 2010. p. 2990–2994.

HASAN, S.M.K.; SARKER, R.; ESSAM, D. Evolutionary scheduling with rescheduling option for sudden machine breakdowns. 2010, [S.l: s.n.], 2010. p. 1–8.

HOLLAND, John. Adaptation in natural and artificial systems: an introductory analysis with application to biology. **Control and artificial intelligence**, 1975.

HOLLAND, John Henry. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. [S.l.]: MIT press, 1992.

HOSSEINABADI, Ali Asghar Rahmani et al. Extended genetic algorithm for solving open-shop scheduling problem. **Soft computing**, v. 23, n. 13, p. 5099–5116, 2019.

KATO, E.R.R.; MORANDIN, O.; FONSECA, M.a.S. **Ant colony optimization algorithm for reactive production scheduling problem in the job shop system**. **2009 IEEE International Conference on Systems, Man and Cybernetics**. [S.l: s.n.]. , 2009

KATO, Edilson R R; MORANDIN, O; FONSECA, M A S. A Max-Min Ant System modeling approach for production scheduling in a FMS. 2010, [S.l: s.n.], 2010. p. 3977–3982.

KAZEMI, Asadollah; MOHAMED, Azah; SHAREEF, Hussain; ZAYANDEHROODI, Hadi. An Improved Power Quality Monitor Placement Method Using MVR Model and Combine Cp and Rp Statistical Indices. **Journal of Electrical Review**, p. 205–209, 2012.

KUNDAKCI, Nilsen; KULAK, Osman. Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. **Computers & Industrial Engineering**, v. 96, p. 31–51, 2016.

KURDI, Mohamed. An effective new island model genetic algorithm for job shop scheduling problem. **Computers & operations research**, v. 67, p. 132–142, 2016.

LU, Hui; SHI, Jinhua; et al. Analysis of the similarities and differences of job-based scheduling problems. **European Journal of Operational Research**, v. 270, n. 3, p. 809–825, 2018.

LU, Po-Hsiang; WU, Muh-Cherng; et al. A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. **Journal of Intelligent Manufacturing**, v. 29, n. 1, p. 19–34, 2018.

MORANDIN, O.; KATO, E.R.R.; et al. **A search method using Genetic Algorithm for production reactive scheduling of manufacturing systems**. **2008 IEEE International Symposium on Industrial Electronics**. [S.l: s.n.]. , 2008

MORANDIN, O.; SANCHES, D.S.; et al. **An Adaptive Genetic Algorithm based approach for production reactive scheduling of manufacturing systems**. **2008 34th Annual Conference of IEEE Industrial Electronics**. [S.l: s.n.]. , 2008

NAQVI, Shaista et al. Transgenic multivitamin corn through biofortification of endosperm with three vitamins representing three distinct metabolic pathways. **Proceedings of the National Academy of Sciences**, v. 106, n. 19, p. 7762–7767, 2009.

NGUYEN, Su; ZHANG, Mengjie; TAN, Kay Chen. Adaptive charting genetic programming for dynamic flexible job shop scheduling. 2018, [S.l: s.n.], 2018. p. 1159–1166.

NIE, Li; GAO, Liang; LI, Peigen; LI, Xinyu. A GEP-based

reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. **Journal of Intelligent Manufacturing**, p. 1–12, 2012.

ÖZTOP, Hande et al. Iterated greedy algorithms for the hybrid flowshop scheduling with total flow time minimization. 2018, [S.l: s.n.], 2018. p. 379–385.

PENG, Chao et al. Analysis of double-resource flexible job shop scheduling problem based on genetic algorithm. 2018, [S.l: s.n.], 2018. p. 1–6.

ROMERO, Miguel A Fernández et al. A heuristic algorithm based on tabu search for the solution of flexible job shop scheduling problems with lot streaming. 2018, [S.l: s.n.], 2018. p. 285–292.

VELDHUIZEN, David A Van; LAMONT, Gary B. On measuring multiobjective evolutionary algorithm performance. 2000, [S.l: s.n.], 2000. p. 204–211.