



A New Genetic Improvement Operator Based on Frequency Analysis for Genetic Algorithms Applied to Job Shop Scheduling Problem

Monique Simplicio Viana¹(✉) , Rodrigo Colnago Contreras² ,
and Orides Morandin Junior¹ 

¹ Federal University of São Carlos, São Carlos, Brazil
{monique.viana, orides}@ufscar.br

² University of São Paulo, São Carlos, Brazil
contreras@usp.br

Abstract. Many researchers today are using meta-heuristics to treat the class of problems known in the literature as Job Shop Scheduling Problem (JSSP) due to its complexity since it consists of combinatorial problems and it is an NP-Hard computational problem. JSSPs are a resource allocation issue and, to solve its instances, meta-heuristics as Genetic Algorithm (GA) are widely used. Although the GAs present good results in the literature, it is very common for these methods that they are stagnant in solutions that are local optima during their iterations and that have difficulty in adequately exploring the search space. To circumvent these situations, we propose in this work the use of an operator specialized in conducting the GA population to a good exploration: the Genetic Improvement based on Frequency Analysis (GIFA). GIFA makes it possible to manipulate the genetic material of individuals by adding characteristics that are believed to be important, with the proposal of directing some individuals who are lost in the search space to a more favorable subspace without breaking the diversity of the population. The proposed GIFA is evaluated considering two different situations in well-established benchmarks in the specialized JSSP literature and proved to be competitive and robust compared to the methods that represent the state of the art.

Keywords: Evolutionary Algorithm · Genetic Algorithm · Genetic improvement · Job Shop Scheduling Problem · Combinatorial optimization

1 Introduction

Combinatorial optimization problems (COPs) consist of situations in which it is necessary to determine, through permutations of elements of a finite set, the configuration of parameters that is more advantageous [26]. Due to its high degree of applicability, many researchers have been using COPs in different contexts. As an example, applications in the logistics [29], vehicle routing [15], railway transport control [22], among other current problems [10]. In particular, one of the most addressed COPs in the literature is production scheduling [27], which, according to Groover [12], is part of the Production

Planning and Control activities and is responsible for determining the design of operations that will be carried out, such as: the environment in which products are processed, what resources are used and what is the start and end time for each production order.

Academic research and the development of solution methodologies have focused on a limited number of classic production scheduling problems, one of the most researched is the variation known as *Job Shop Scheduling Problem (JSSP)* [14], in which a finite set of jobs must be processed by a finite set of machines. In this category of problems, the objective is usually to determine a configuration in the order of processing of a set of jobs, or tasks, to minimize, for example, the time of using resources [39]. In this case, several performance measures are useful to evaluate how satisfactory a given configuration is for a JSSP, with *makespan* [38], which corresponds to the total time needed to finish the production of a set of jobs, one of the most used.

Belonging to the well-known class of problems *NP-Hard*, JSSP presents itself as a computational challenge, since it is not a trivial task to develop an approach to determine exact solutions that represent a configuration with an adequate performance measure, within a reasonable time, even considering small and moderate cases [35]. From this need, algorithms that present approximate results in a feasible computational time were developed and applied to JSSP. The main methods used are those composed of meta-heuristics [23], mainly by the Evolutionary Algorithm (EA) known as Genetic Algorithm (GA) [21, 24, 31–33]. Even so, the JSSP consists of a class of problems that remain open [6] and with many instances still unsolved in the well-known benchmarks of the area [9]. This is because the existing methods do not have the necessary efficiency to guarantee their practical use.

More specifically, it is possible to highlight some disadvantages in the use of GA in solving COPs [4, 8]. In detail, it is common for this set of techniques that they become stagnant [30], during their iterations, in solutions that are local minimums, which configures the phenomenon known as premature convergence [41]. Also, GAs may require high computational time [20] to obtain good solutions to this type of problem. Therefore, for complex problems, GA needs to be assimilated to specific problem routines to make the approach effective. Hybridization can be a deeply effective way to improve the performance of these techniques. The most common form of hybridization is the addition of GAs to local search strategies and the incorporation of domain-specific knowledge in the search process [28]. In the latter, there are genetic improvement operators through manipulations in specific genes on a chromosome. These have as main objective to provide to individuals who are not able to stand out in a population the reinforcement coming from one or more individuals who have been successful in the adaptation process. In other words, these operators direct the worst individuals in a population to areas known to be good in the search space.

The authors do Amaral and Hruschka JR [2, 3] present an operator in this line of reasoning, entitled transgenic operator and that simulates the process of genetic improvement. To conduct such a procedure, in one of the stages of the GA, the population of the same is replicated to four parallel sub-populations and, in each of these four populations, the best individuals transfer up to 4 genes, based on historical information, to selected individuals. Then, only the best individuals among the four sub-populations remains. Viana, Morandin Junior and Contreras [33] proposed an adaptation of the

transgenic operator do Amaral and Hruschka JR [3] to solve a JSSP with GA. The authors propose the identification of relevance in the genes used in the transgenic process through a pre-processing process. However, such preprocessing is computationally time-consuming and may not be viable in large JSSPs.

In this work, we propose a new population guidance operator for GAs: the Genetic Improvement based on Frequency Analysis (GIFA) Operator. Our method consists of a new way to determine the genetic relevance based on the frequency analysis of the genes of individuals who have good fitness values in the population. We also propose the construction of a representative individual that represents this group of good individuals and that it is used in the process of genetic manipulation to guide the worst individuals towards good solutions and, possibly, that these become positive highlights in the population.

This work is divided into five sections. Specifically, in Sect. 2, we describe the JSSP basis. We present, in Sect. 3, the details about the proposed GIFA operator and the requirements that an GA needs to satisfy to use it. Experimental results on different GAs using GIFA and the advancement in the state of the art of JSSPs are presented in Sect. 4. The work is finished in Sect. 5 with conclusions about the developments carried out and future projections for improving the method and possible applications.

2 Formulation of Job Shop Scheduling Problem

We can define JSSP as a COP that has a set of N jobs that must be processed on a set of M machines. Also, each job has a script that determines the order in which it must pass through the machines for its process to be completed. Each job processing per machine represents an operation and the objective of a JSSP can be interpreted as being the challenge of determining the optimal sequencing of operations with one or more performance measures as a guide. The components of this problem follow some restrictions [39]:

- Each job can be processed on a single machine at a time;
- Each machine can process only one job at a time;
- Operations are considered non-preemptive, i.e., cannot be interrupted,
- Configuration times are included in processing times and are independent of sequencing decisions.

In this work, we adopted makespan (MKS) as a performance measure. The MKS is the total time that a JSSP instance takes to complete the processing of a set of jobs on a set of machines taking into account a given operation sequence.

Mathematically, let's assume the following components of a JSSP:

- $J = \{J_1, J_2, \dots, J_N\}$ is the set of jobs;
- $\mathcal{M} = \{m_1, m_2, \dots, m_M\}$ is the set of machines;
- $O = (O_1, O_2, \dots, O_{N \cdot M})$ is a operation sequence that sets the priority order for processing the set of jobs in the set of machines
- $T_i(O)$ represents the time taken by the job J_i to be processed by all machines in its script according to the operation sequence defined in O .

Then, according to [7], the MKS can be defined as the total time that all jobs take to be processed according to a given operation sequence, as presented in Eq. (1).

$$\text{MKS} = \max_i T_i(O). \tag{1}$$

3 A New Genetic Improvement Operator Based on Frequency Analysis for GA Applied to JSSP

In this section, we will present in detail how the proposed method works. We will specify the idea of determining genetic relevance by analyzing the frequency of genes that represent good characteristics in individuals with adequate fitness values in the population and, with that, we intend to obtain innovation with the following three topics:

- A new strategy for defining genetic relevance in GAs chromosomes;
- A new genetic improvement operator that is versatile and can be used in GAs variations,
- Improving the state of the art of JSSP benchmark results.

3.1 Genetic Representation

Our operator was developed to operate in all GA-like methods with minor modifications. In the meantime, we are going to conduct its fundamentation on a specific encoding. In this case, we will use the “coding by operation order” [5]. In this representation [32], the feasible space of a JSSP instance defined by N jobs and M machines is formed by chromosomes $c \in \mathbb{N}^{N \cdot M}$, such that exactly M coordinates of c are equal to i (representing the job index i), for every $i \in \{1, 2, \dots, N\}$.

This encoding determines in chromosome the operation priority with respect to machine allocation. For example [31], let’s assume $c = (2, 1, 2, 2, 1, 1)$ as being a feasible solution in a JSSP instance with dimension 2×3 ($N = 2$ and $M = 3$). Thus, according to the operations defined in c , the following actions must be carried out in parallel or if the previous action has already been done.:

- 1st) Job 2 must be processed by the 1st machine of its script.
- 2nd) Job 1 must be processed by the 1st machine of its script.
- 3rd) Job 2 must be processed by the 2nd machine of its script.
- 4th) Job 2 must be processed by the 3rd machine of its script.
- 5th) Job 1 must be processed by the 2nd machine of its script.
- 6th) Job 1 must be processed by the 3rd machine of its script.

3.2 Fitness Function

The encoding used makes it natural to define the fitness function of the problem as the makespan of a JSSP instance given according to the stipulated operation sequence. That is, the fitness function [33] used is given according to Eq. (2):

$$\begin{aligned} F : \mathbb{O} &\longrightarrow \mathbb{R} \\ O &\longmapsto F(O) := \max_i T_i(O), \end{aligned} \tag{2}$$

in which \mathbb{O} is the set of all possible operation sequences for the defined JSSP instance.

In this way, for this fitness function, the MKS of the JSSP instance is calculated according to a given operation sequence, then the meta-heuristic must look for an operation sequence in which the MKS is as small as possible and, consequently, the set of jobs must be processed by the set of machines taking the shortest possible time.

3.3 Proposed Genetic Improvement Based on Frequency Analysis Operator

In this work, we propose a new genetic improvement operator for evolutionary algorithms: the GIFA operator. The operator is based on a frequency analysis matrix calculated during the iterations of each GA. GIFA aims to calculate which genes on a chromosome can direct individuals with poor fitness values to better solutions and better search spaces. GIFA has two main stages: the first being defined by the making of the representative individual, that is, an individual that is determined by the configuration of the most frequent genes in the best individuals in the population; and the second stage consists of the use of the representative individual in the transgenic process, that is, the genetic manipulation through the insertion of specific genes of the representative individual in genes of the worst individuals in the population. Below, we present these steps in detail.

Stage 1: Composition of the Representative Individual. Initially, a portion of the population that presents the best fitness values is selected. Specifically, we select N_{Top} individuals who are considered good examples of solutions in the population. Then, for each index job i , a frequency vector $\vec{v}_i \in \mathbb{R}^{N \cdot M}$ is associated, in which the number of its occurrences is stored in each coordinate where the product i appears exactly at the position of this coordinate on the chromosomes selected for comparison. In Fig. 1, an example of the calculation of the frequency vectors \vec{v}_i is presented when considering 4 individuals c_1, c_2, c_3 , and c_4 with the best values of fitness in a 3×2 dimension JSSP instance.

Once the vector \vec{v}_i has been made for every job index i , a chromosome whose coordinates are determined by the job with the highest frequency in this coordinate is defined as a representative individual. That is, each gene (coordinate) of the representative individual is defined as the job index that is most present in this coordinate in the best individuals in the population. It is also possible to establish an order of genetic relevance according to the frequency vectors \vec{v}_i . That is, it is possible to define which genes of the representative individual are more suitable to be transferred in the process of genetic improvement. Such relevance is also defined according to the frequency that the products present in each coordinate of the best individuals so that the genes that present the same job in many good individuals can categorize a “trend” that leads to good fitness values. Therefore, these genes must be considered to be relevant, since they describe a positive characteristic in several individuals that stand out in the population. Mathematically, the representative individual and its genetic relevance are made according to the following procedure:

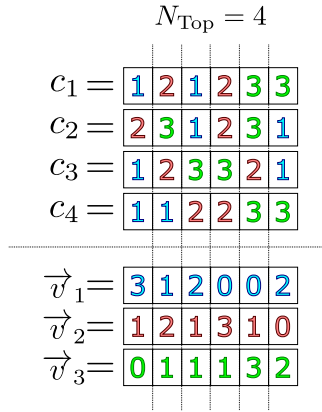


Fig. 1. Calculating the frequency vectors (\vec{v}_i) of the three jobs in each coordinate of the four best chromosomes in the population.

1. Let \mathbf{c} be the representative individual and w a vector that designates a score for each of its coordinates, initially null. In the following items, the coordinates of \mathbf{c} and w are made.
2. We define I_1 as being $\arg \max_i \{\vec{v}_{i,1}\}$. That is, I_1 is the index of the job that has the highest frequency in the first coordinate of the exemplary individuals. Therefore, the first coordinate of the representative individual is defined as I_1 . Mathematically,

$$\mathbf{c}_1 := I_1.$$

In addition, a w_1 score defined as the maximum frequency shown in the first coordinate of the best individuals is associated with the first coordinate of \mathbf{c} . That is,

$$w_1 := \max_i \{\vec{v}_{i,1}\} = \vec{v}_{I_1,1}.$$

3. Assign the value 2 to j .
4. We define I_j as being $\arg \max_i \{\vec{v}_{i,j}\}$, that is, I_j is the most frequent product index in the j coordinate in the N_{Top} individuals. However, in order to guarantee the feasibility of the representative individual, it is necessary to establish two more restrictions:
 - 4.1 If the product I_j is not in M coordinates of \mathbf{c} , then it is defined as I_j the j -th coordinate of the representative individual. That is,

$$\mathbf{c}_j := I_j.$$

In this case, the respective score is associated with the j -th coordinate of the representative individual as the maximum possible value presented in the j -th coordinate of the best individuals. That is,

$$w_j := \max_i \{\vec{v}_{i,j}\} = \vec{v}_{I_j,j}.$$

4.2 Otherwise, to guarantee the feasibility of \mathbf{c} , the frequencies of the index job I_j are disregarded, since it is already arranged in M coordinates of \mathbf{c} and, therefore, does not can occupy any more of its coordinates. To do so, we must cancel its respective frequency vector, that is,

$$\vec{v}_{I_j} := \vec{0}.$$

To make a new attempt, we must return to item 4.

5. If $j \neq N \cdot M$ then $j := j + 1$ and we must return to item 4. Otherwise, the procedure is finished and we have the representative individual pair and its respective genetic score (\mathbf{c}, w) .

Note that it is not necessary to project the representative individual in the feasible space of the problem since due to its construction and the item 4 above, it is already feasible. In Fig. 2, an example of the calculation of the representative individual (\mathbf{c}) and the relevance of its genes (w) in a JSSP instance of dimension 4×3 is presented, taking as best individuals the $N_{\text{Top}} = 5$ individuals with the lowest fitness values available in the population.

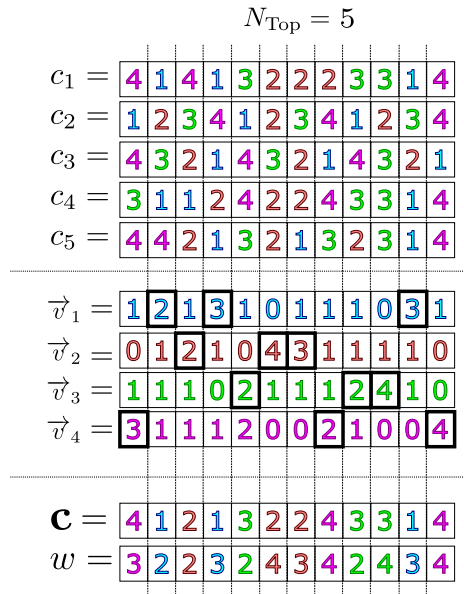


Fig. 2. Computation of the representative individual (\mathbf{c}) and its genetic relevance (w).

Stage 2: Use of the Representative Individual in Genetic Improvement. Once the representative individual and the relevance of each of its genes have been calculated, then it is proposed that its most relevant genes be transferred to the worst individuals in

the population, thus simulating a mechanism for genetic improvement, or transgenics. For this, we take $P_{\text{Worst}} := \{x_1, x_2, \dots, x_{N_{\text{Worst}}}\}$ as the set of the worst N_{Worst} individuals in a population. Subsequently, the most significant, or most relevant, N_{Genes} genes of the representative individual are transferred to all individuals in the P_{Worst} maintaining their original positions. This procedure can generate infeasible solutions. Thus, it is necessary to conduct a correction, or projection, process on the individuals resulting from this operation. For this, we carry out the projection through the Hamming distance [37] modifying only the genes that were not received from the representative individual. In this way, the individuals generated in this procedure are projected on the feasible set of the problem, giving rise to the genetically improved individuals $P_{\text{Improved}} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_{N_{\text{Worst}}}\}$.

It is also necessary to establish how many genes will be transferred from the representative individual to the individuals of P_{Worst} . For this, we will follow a procedure similar to that of Viana, Morandin Junior, and Contreras [33], which empirically determine that the adequate amount of genes used in the genetic improvement process is given by the root of the number of coordinates of the chromosome. Thus, the process remains advantageous and does not cause early convergence in the population. Thus, in this work, N_{Genes} is defined as $\text{round}(\sqrt{N \cdot M})$. In Fig. 3, an example of the determination of the most significant genes of a representative individual \mathbf{c} when it is given the scores of his genes w while addressing a JSSP with dimension 4×3 .

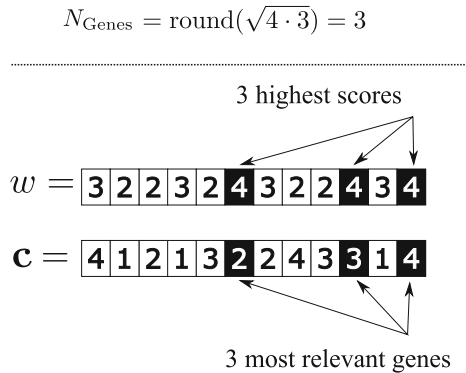


Fig. 3. Determination of the most significant genes of a representative individual.

Assuming $N_{\text{Worst}} = 3$ and $P_{\text{Worst}} = \{x_1, x_2, x_3\}$ as the set of the worst 3 individuals in a population, the improvement process is shown in Fig. 4 genetic that transfers the N_{Genes} best genes from the representative individual \mathbf{c} of Fig. 4 to all individuals in the set P_{Worst} .

The genetic improvement procedure must be performed after the standard operators of the GA, or the GA-like method used, and right after the generation of a new population. Thus, the set P_{Worst} must be formed by individuals from the new population of the method. Besides, after the application of the genetic improvement, the evaluation of improvement or worsening of the affected individuals is made, so that the genetic

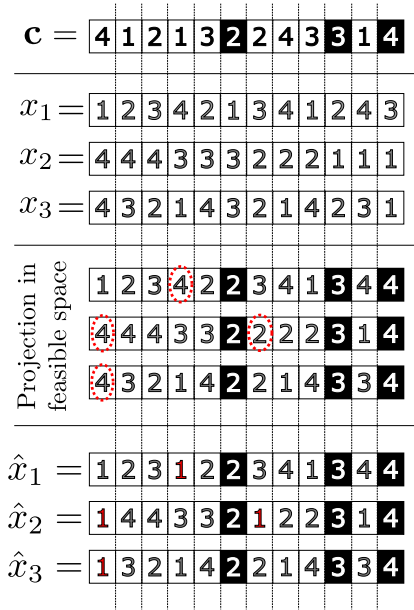


Fig. 4. Genetic improvement proposed. The genes highlighted on a black background are the most relevant, while the genes highlighted with the red sectioned circle are those that need correction.

changes made will only be saved in individuals who have obtained an improvement in fitness. That is, only individuals who have gained an advantage in the process of genetic improvement will be substituted in the population; the other individuals should be discarded and replaced by new individuals generated randomly.

3.4 Scheme of Use for Proposed Operators: Algorithm Structure

The proposed genetic improvement strategy was developed to be as versatile as possible in the sense that it can be attached to any GA-like method. Thus, the proposed operator must be used after the execution of the original operators of the method considered in order to guide solutions that were not able to stand out through the traditional strategies defined in the method. In other words, to use the proposed operator in a given GA-like method, we must obey the following steps:

1. Define the initial parameters and specifics of the chosen GA-like method.
2. Execute the operators that make up the GA-like method. These being, for example, the operators of crossover, mutation, local search, creation of new population, etc.
3. At the end of an iteration involving the traditional operators of the selected GA-like method, we will make a sub-population P_{Worst} with the worst N_{Worst} individuals in the current population.
4. At the same time, we will select the best N_{Top} individuals in the population to make up the representative individual.

5. Build the representative individual using the strategy described in **Stage 1** of the Sect. 3.3.
6. Determine a relevance scale to the genes of the representative individual.
7. Conduct the genetic improvement of the P_{Worst} individuals using the most relevant N_{Genes} genes of the representative individual.
8. Replace in the current population of the method all individuals who obtained an improvement in the fitness value in the process of genetic improvement and return in the execution of the original operators of the considered GA-like method. Those who have not improved should be replaced by new individuals randomly generated according to Levy's exponential distribution, following the procedure of Al-Obaidi and Hussein [1].

In Fig. 5, we present a flowchart that illustrates the sequence of steps of the proposed genetic improvement process.

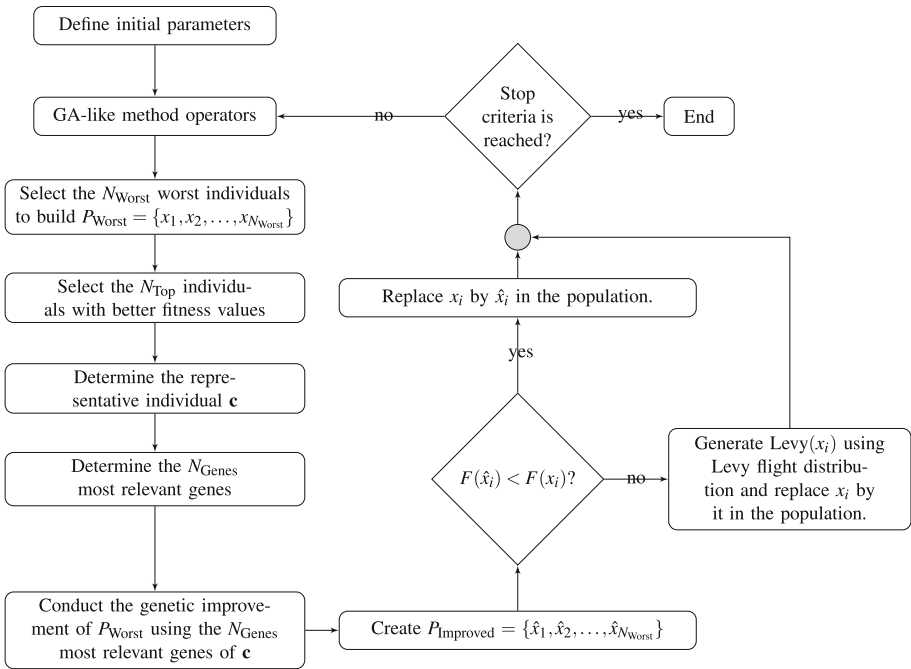


Fig. 5. Flow chart of our proposed Genetic Improvement operator for Genetic Algorithm.

4 Implementation and Experimental Results

4.1 Experimental Environment

For the conduction of the experiments, we considered two different situations: in the first, we evaluated the impact that the proposed operator causes on five GA-like methods, all of which were obtained using the framework of Viana, Morandin Junior and

Contreras [32], in three JSSP instances of varying complexity; in the second, we compare with recent methods in the literature the ability of the proposed operator to look for good solutions in 43 instances of JSSP that make up the area benchmark, with 3 from Fisher and Thompson (FT) [11] and 40 from Lawrence (LA) [19]. In detail, in this second situation, we consider relevant and recent methods which deal with the JSSP with the same specific instances and, when existing, presented in papers published in the last three years. In all, we consider for comparison the following methods: mXLSGA [32], NGPSO [40], SSS [13], GA-CPG-GT [18], DWPA [34], GWO [16], IPB-GA [17] and aLSGA [4]. The proposed algorithm is coded in MATLAB and we performed the evaluations on a computer with 2.4 GHz Intel(R) Core i7 CPU and 16 GB of RAM.

4.2 Results and Comparison with Other Algorithms

For the first testing situation, we will consider five variations of the Viana, Morandin Junior and Contreras [32] framework: a basic GA (GA), GA with Search Area Adaptation (GSA) [36], GA with Local Search (LSGA) [25], GA with Elite Local Search and agent adjustment (aLSGA) [4], and GA with multi-crossover and massive local search (mXLSGA) [32]. In each of these versions, we added the proposed genetic enhancement operator, GIFA, and conducted our evaluations on three JSSP instances: FT 06, with a dimension of 6×6 , and the best-known solution (BKS) equal to 55; LA 01, with a dimension of 10×5 , and BKS equal to 666; and LA 16, with a dimension of 10×10 , and BKS equal to 945. Thus, each GA-like method considered has a version with the proposed operator, represented by the acronym GIFA together with its standard acronym. Our main purpose in this situation is to evaluate the impact of using GIFA in each of the GA-like methods, so we kept the best possible configuration of each of the methods available in the original works, with the exception that everyone had 100 individuals in their populations and run for 100 generations. In addition, we added to each of them the configuration referring to GIFA, which is defined as follows: $N_{\text{Top}} = N_{\text{Worst}} = 10$. In this case, the best value, the worst value, the mean, and the standard deviation (SD) of the makespan values calculated at 35 independent executions of each method on the three JSSP instances considered are presented in Table 1.

Looking at Table 1, we noticed that the operator made all methods more stable, reducing the magnitude of the worst makespan value found, the mean and standard deviation of all of them in all situations where it was possible to have improvement. However, in the most complex instance, the LA 16, our operator was able to improve the best makespan value only in the case of the aLSGA technique. This serves as an indication that the proposed operator brings a considerable increase in the stability of the method, but the ability to explore the search space still has a strong dependence on the original technique used. This is because our GIFA guides the population towards guiding individuals with bad makespan values in regions where individuals with good fitness values are known to increase local exploration and, therefore, find good solutions, but it is up to the base technique to indicate good search regions.

Thus, the second situation considered should serve as an experiment in this sense, so that we can evaluate the ability of the proposed operator to increase the search and exploration power of a given technique. For this, we will add the proposed GIFA operator in a technique already known to be effective in finding good solutions in the JSSP

Table 1. GA-like methods statistics for 35 executions of each method.

Instance	Method	Best	Worst	Average	SD
FT 06	GA	55	57	55.45	0.85
	GIFA-GA	55	56	55.14	0.21
	GSA	55	55	55	0
	GIFA-GSA	55	55	55	0
	LSGA	55	59	57.68	1.43
	GIFA-LSGA	55	56	55.84	0.73
	aLSGA	55	55	55	0
	GIFA-aLSGA	55	55	55	0
	mXLSGA	55	55	55	0
	GIFA-mXLSGA	55	55	55	0
LA 01	GA	666	712	679.02	9.98
	GIFA-GA	666	678	669.37	5.17
	GSA	666	715	677.8	13.61
	GIFA-GSA	666	687	672.34	7.43
	LSGA	666	726	697	16.65
	GIFA-LSGA	666	707	688.67	15.59
	aLSGA	666	666	666	0
	GIFA-aLSGA	666	666	666	0
	mXLSGA	666	666	666	0
	GIFA-mXLSGA	666	666	666	0
LA 16	GA	982	1100	1045.6	26.40
	GIFA-GA	982	1061	1022.89	20.51
	GSA	994	1110	1046.77	26.37
	GIFA-GSA	994	1021	1017.38	15.49
	LSGA	1016	1148	1084.25	32.27
	GIFA-LSGA	1016	1077	1037.11	26.62
	aLSGA	959	985	980.51	4.48
	GIFA-aLSGA	956	982	975.12	2.36
	mXLSGA	945	982	972.25	13.30
	GIFA-mXLSGA	945	979	959.93	6.37

instances that make up the benchmark today: the mXLSGA [32]. In this case, we will evaluate GIFA-mXLSGA at 40 instances LA and 3 FT instances. In Table 2, we presented the results derived from 10 independent executions of our method on the LA and FT instance tests. The columns indicate, respectively, the instance that was tested, the instance size (number of Jobs \times number of Machines), the optimal solution of each instance, the results achieved by each method considering all the executions (best solu-

tion found and error percentage (Eq. (3)), and the mean of the error with respect to each benchmark (MErr).

$$E_{\%} = 100 \times \frac{\text{Best} - \text{BKS}}{\text{BKS}}, \quad (3)$$

in which $E_{\%}$ is the relative error, “BKS” is the best known Solution and “Best” is the best value obtained by executing the algorithm 10 times for each instance.

Analyzing Table 2, we can verify that the GIFA operator was able to improve the search capability of the mXLSGA method. Specifically, considering only the LA instances, the use of the proposed operator was able to reduce the magnitude of the average relative error by 0.12, which corresponds to a reduction of 19.67% of its value. In other words, the GIFA operator made the mXLSGA method able to find the best-known makespan in 72.5% of LA instances, obtaining an average relative error of 0.49, the lowest of all methods. Concerning FT instances, the proposed GIFA operator did not compromise the search capability of mXLSGA, causing the best-known solutions to be found in all instances. In summary, we can highlight some points when analyzing the results referring to Table 2:

- There was no worsening of the results in any instance with the use of the proposed operator;
- The GIFA-mXLSGA method obtained the lowest $E_{\%}$;
- The proposed operator made mXLSGA able to find the BKS in the LA 22 instance,
- The proposed operator improved the results of mXLSGA by 7 LA instances.

With the results of the two situations considered, we note that the proposed method is effective in increasing the stability and efficiency of finding good solutions for GA-like methods.

5 Conclusion

The objective of this work was to develop a new GA-like method operator to minimize the makespan in JSSP instances. The proposed technique was a new genetic improvement operator based on a new frequency analysis strategy to detect relevance in genes, titled GIFA operator. To evaluate the proposed approach, experiments were conducted in 43 JSSP instances of varying complexity. The instances used were FT [11] and LA [19]. The results obtained were compared with other approaches in related works: mXLSGA [32], NGPSO [40], SSS [13], GA-CPG-GT [18], DWPA [34], GWO [16], IPB-GA [17] and aLSGA [4].

We evaluate the potential of the proposed operator in two analysis situations. In the first, we involved the use of GIFA in five different GA-like methods, which were simulated by the framework of Viana, Morandin Junior and Contreras [32], and in all cases, the operator increased the stability of the technique considered improving the mean and standard deviation of the solutions found in three instances of JSSP. In the second, the performance of the GIFA used in the mXLSGA [32] was compared with methods that make up the state of the art in the specialized literature and we note that the use of the proposed operator was effective in increasing the search capacity of the

Table 2. Comparison of computational results between mXLSGA and other algorithms. The symbol “-” means “no evaluated in that instance”.

Instance	Size	BKS	GIFA-mXLSGA		mXLSGA		NGPSO		SSS		GA-CPG-GT		DWPA		GWO		IPB-GA		aLSGA			
			Best	E%	Best	E%	Best	E%	Best	E%	Best	E%	Best	E%	Best	E%	Best	E%	Best	E%	Best	E%
LA01	10 × 5	666	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00	666	0.00
LA02	10 × 5	655	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00	655	0.00
LA03	10 × 5	597	597	0.00	597	0.00	597	0.00	597	0.00	597	0.00	614	2.84	597	0.00	599	0.33	606	1.50		
LA04	10 × 5	590	590	0.00	590	0.00	590	0.00	590	0.00	590	0.00	598	1.35	590	0.00	590	0.00	593	0.50		
LA05	10 × 5	593	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00	593	0.00
LA06	15 × 5	926	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00	926	0.00
LA07	15 × 5	890	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00	890	0.00
LA08	15 × 5	863	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00	863	0.00
LA09	15 × 5	951	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00	951	0.00
LA10	15 × 5	958	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00	958	0.00
LA11	20 × 5	1222	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00	1222	0.00
LA12	20 × 5	1039	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00	1039	0.00
LA13	20 × 5	1150	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00	1150	0.00
LA14	20 × 5	1292	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00	1292	0.00
LA15	20 × 5	1207	1207	0.00	1207	0.00	1207	0.00	1207	0.00	1207	0.00	1273	5.46	1207	0.00	1207	0.00	1207	0.00	1207	0.00
LA16	10 × 10	945	945	0.00	945	0.00	945	0.00	947	0.21	946	0.10	993	5.07	956	1.16	946	0.10	946	0.10	946	0.10
LA17	10 × 10	784	784	0.00	784	0.00	794	1.27	784	0.00	794	0.00	793	1.14	790	0.76	784	0.00	784	0.00	784	0.00
LA18	10 × 10	848	848	0.00	848	0.00	848	0.00	848	0.00	848	0.00	861	1.53	859	1.29	853	0.58	848	0.00		
LA19	10 × 10	842	842	0.00	842	0.00	842	0.00	842	0.00	842	0.00	888	5.46	845	0.35	866	2.85	852	1.18		
LA20	10 × 10	902	902	0.00	902	0.00	908	0.66	907	0.55	907	0.55	934	3.54	937	3.88	913	1.21	907	0.55		
LA21	15 × 10	1046	1052	0.57	1059	1.24	1183	13.09	1076	2.86	1090	4.20	1105	5.64	1090	4.20	1081	3.34	1068	2.10		
LA22	15 × 10	927	927	0.00	935	0.86	927	0.00	954	2.91	954	2.91	989	6.68	970	4.63	970	4.63	956	3.12		
LA23	15 × 10	1032	1032	0.00	1032	0.00	1032	0.00	1032	0.00	1032	0.00	1051	1.84	1032	0.00	1032	0.00	1032	0.00	1032	0.00
LA24	15 × 10	935	940	0.53	946	1.17	968	3.52	974	4.17	974	4.17	988	5.66	982	5.02	1002	7.16	966	3.31		
LA25	15 × 10	977	984	0.71	986	0.92	977	0.00	999	2.25	999	2.25	1039	6.34	1008	3.17	1023	4.70	1002	2.55		
LA26	20 × 10	1218	1218	0.00	1218	0.00	1218	0.00	1237	1.55	1237	1.55	1303	6.97	1239	1.72	1273	4.51	1223	0.41		
LA27	20 × 10	1235	1261	2.10	1269	2.75	1394	12.87	1256	1.70	1313	6.31	1346	8.98	1290	4.45	1317	6.63	1281	3.72		
LA28	20 × 10	1216	1239	1.89	1239	1.89	1216	0.00	1280	5.26	1280	5.26	1291	6.16	1263	3.86	1288	5.92	1245	2.38		
LA29	20 × 10	1152	1190	3.29	1201	4.25	1280	11.11	1247	8.24	1247	8.24	1275	10.67	1244	7.98	1233	7.03	1230	6.77		
LA30	20 × 10	1355	1355	0.00	1355	0.00	1355	0.00	1367	0.88	1367	0.88	1389	2.50	1355	0.00	1377	1.62	1355	0.00		
LA31	30 × 10	1784	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00	1784	0.00
LA32	30 × 10	1850	1850	0.00	1850	0.00	1850	0.00	1850	0.00	1850	0.00	1850	0.00	1850	0.00	1851	0.05	1850	0.00		
LA33	30 × 10	1719	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00	1719	0.00
LA34	30 × 10	1721	1721	0.00	1721	0.00	1721	0.00	1725	0.23	1725	0.23	1788	3.89	1721	0.00	1749	1.62	1721	0.00		
LA35	30 × 10	1888	1888	0.00	1888	0.00	1888	0.00	1888	0.00	1888	0.00	1947	3.125	1888	0.00	1888	0.00	1888	0.00	1888	0.00
LA36	15 × 15	1268	1295	2.12	1295	2.12	1408	11.04	1304	2.83	1308	3.15	1388	9.46	1311	3.39	1334	5.20	-	-		
LA37	15 × 15	1397	1407	0.71	1415	1.28	1515	8.44	1489	6.58	1489	6.58	1486	6.37	-	-	1467	5.01	-	-		
LA38	15 × 15	1196	1246	4.18	1246	4.18	1196	0.00	1275	6.60	1275	6.60	1339	11.95	-	-	1278	6.85	-	-		
LA39	15 × 15	1233	1258	2.02	1258	2.02	1662	34.79	1290	4.62	1290	4.62	1334	8.19	-	-	1296	5.10	-	-		
LA40	15 × 15	1222	1243	1.71	1243	1.71	1222	0.00	1252	2.45	1252	2.45	1347	10.22	-	-	1284	5.07	-	-		
MErr			0.49		0.61		2.42		0.59		1.50		3.52		1.27		1.99		0.80			
FT06	6 × 6	55	55	0.00	55	0.00	55	0.00	55	0.00	55	0.00	-	-	55	0.00	55	0.00	55	0.00	55	0.00
FT10	10 × 10	930	930	0.00	930	0.00	930	0.00	936	0.64	935	0.53	-	-	940	1.07	960	3.22	930	0.00		
FT20	20 × 5	1165	1165	0.00	1165	0.00	1210	3.86	1165	0.00	1180	1.28	-	-	1178	1.11	1192	2.31	1165	0.00		
MErr			0.00		0.00		1.28		0.21		0.60		-		0.73		1.84		0.00			

mXLSGA, since GIFA-mXLSGA was the method with the lowest average relative error among all the techniques considered. Thus, we conclude that the proposed operator

was able to achieve the stipulated objective since it statistically directed the GA-like methods evaluated for search spaces with better solutions.

In future works, we will make use of deep learning techniques to detect relevance during GAs iterations through reinforcement learning approaches, which should make the methodology more robust and accurate. Also, we will add assessments about processing time measurements and expand the methodology to a greater number of instances and benchmarks.

Acknowledgments. This study was financed in part by the “*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil*” (CAPES) - Finance Code 001, and by the Brazilian National Council for Scientific and Technological Development, process #381991/2020-2.

References

1. Al-Obaidi, A.T.S., Hussein, S.A.: Two improved cuckoo search algorithms for solving the flexible job-shop scheduling problem. *Int. J. Percept. Cogn. Comput.* **2**(2), 25–31 (2016)
2. do Amaral, L.R., Hruschka, E.R.: Transgenic, an operator for evolutionary algorithms. In: 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 1308–1314. IEEE (2011)
3. do Amaral, L.R., Hruschka Jr, E.R.: Transgenic: an evolutionary algorithm operator. *Neurocomputing* **127**, 104–113 (2014)
4. Asadzadeh, L.: A local search genetic algorithm for the job shop scheduling problem with intelligent agents. *Comput. Ind. Eng* **85**, 376–383 (2015)
5. Bierwirth, C., Mattfeld, D.C., Kopfer, H.: On permutation representations for scheduling problems. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 310–318. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61723-X_995
6. Çaliş, B., Bulkan, S.: A research survey: review of AI solution strategies of job shop scheduling problem. *J. Intell. Manuf* **26**(5), 961–973 (2015)
7. Chaudhry, I.A., Khan, A.A.: A research survey: review of flexible job shop scheduling techniques. *Int. Trans. Oper. Res.* **23**(3), 551–591 (2016)
8. Contreras, R.C., Morandin Junior, O., Viana, M.S.: A new local search adaptive genetic algorithm for the pseudo-coloring problem. In: Tan, Y., Shi, Y., Tuba, M. (eds.) ICSI 2020. LNCS, vol. 12145, pp. 349–361. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53956-6_31
9. Demirkol, E., Mehta, S., Uzsoy, R.: Benchmarks for shop scheduling problems. *Eur. J. Oper. Res.* **109**(1), 137–141 (1998)
10. Ehrgott, M., Gandibleux, X.: Multiobjective combinatorial optimization—theory, methodology, and applications. In: *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, pp. 369–444. Springer, Heidelberg (2003). https://doi.org/10.1007/0-306-481107-3_8
11. Fisher, C., Thompson, G.: Probabilistic learning combinations of local job-shop scheduling rules. In: *Industrial Scheduling* pp. 225–251 (1963)
12. Groover, M.P.: *Fundamentals of Modern Manufacturing: Materials Processes, and Systems*. John Wiley & Sons, Hoboken (2007)
13. Hamzadayı, A., Baykasoglu, A., Akpınar, Ş: Solving combinatorial optimization problems with single seekers society algorithm. *Knowl.-Based Syst.* **201**, 106036 (2020)
14. Hart, E., Ross, P., Corne, D.: Evolutionary scheduling: a review. *Genetic Program. Evol. Mach.* **6**(2), 191–220 (2005)

15. James, J., Yu, W., Gu, J.: Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst* **20**(10), 3806–3817 (2019)
16. Jiang, T., Zhang, C.: Application of grey wolf optimization for solving combinatorial problems: job shop and flexible job shop scheduling cases. *IEEE Access* **6**, 26231–26240 (2018)
17. Jorapur, V.S., Puranik, V.S., Deshpande, A.S., Sharma, M.: A promising initial population based genetic algorithm for job shop scheduling problem. *J. Softw. Eng. Appl* **9**(05), 208 (2016)
18. Kurdi, M.: An effective genetic algorithm with a critical-path-guided giffler and thompson crossover operator for job shop scheduling problem. *Int. J. Intell. Syst. Appl. Eng* **7**(1), 13–18 (2019)
19. Lawrence, S.: Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Carnegie-Mellon University, Graduate School of Industrial Administration (1984)
20. Li, X., Gao, L.: An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int. J. Prod. Econ.* **174**, 93–110 (2016)
21. Lu, Y., Huang, Z., Cao, L.: Hybrid immune genetic algorithm with neighborhood search operator for the job shop scheduling problem. In: *IOP Conference Series: Earth and Environmental Science*, vol. 474, 052093 (2020)
22. Matyukhin, V., Shabunin, A., Kuznetsov, N., Takmazian, A.: Rail transport control by combinatorial optimization approach. In: *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–4. IEEE (2017)
23. Mhasawade, S., Bewoor, L.: A survey of hybrid metaheuristics to minimize makespan of job shop scheduling problem. In: *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 1957–1960. IEEE (2017)
24. Milovsevic, M., Lukic, D., Durdev, M., Vukman, J., Antic, A.: Genetic algorithms in integrated process planning and scheduling—a state of the art review. *Proc. Manuf. Syst.* **11**(2), 83–88 (2016)
25. Ombuki, B.M., Ventresca, M.: Local search genetic algorithms for the job shop scheduling problem. *Appl. Intell.* **21**(1), 99–109 (2004)
26. Pardalos, P.M., Du, D.-Z., Graham, R.L. (eds.): *Handbook of Combinatorial Optimization*. Springer, New York (2013). <https://doi.org/10.1007/978-1-4419-7997-1>
27. Parente, M., Figueira, G., Amorim, P., Marques, A.: Production scheduling in the context of industry 4.0: review and trends. *Int. J. Prod. Res.* **58**, 1–31 (2020)
28. Sastry, K., Goldberg, D., Kendall, G.: Genetic algorithms. In: Burke, E.K., Kendall, G. (eds.) *Search Methodologies*, pp. 97–125. Springer, Heidelberg (2005). https://doi.org/10.1007/0-387-28356-0_4
29. Sbihi, A., Eglese, R.W.: Combinatorial optimization and green logistics. *Ann. Oper. Res.* **175**(1), 159–175 (2010)
30. Viana, M.S., Morandin Junior, O., Contreras, R.C.: An improved local search genetic algorithm with a new mapped adaptive operator applied to pseudo-coloring problem. *Symmetry* **12**(10), 1684 (2020)
31. Viana, M.S., Junior, O.M., Contreras, R.C.: An improved local search genetic algorithm with multi-crossover for job shop scheduling problem. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) *ICAISC 2020. LNCS (LNAI)*, vol. 12415, pp. 464–479. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61401-0_43
32. Viana, M.S., Morandin Junior, O., Contreras, R.C.: A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem. *Sensors* **20**, 5440 (2020). <https://doi.org/10.3390/s20185440>

33. Viana, M.S., Morandin Junior, O., Contreras, R.C.: Transgenic genetic algorithm to minimize the makespan in the job shop scheduling problem. In: Proceedings of the 12th International Conference on Agents and Artificial Intelligence, vol. 2: ICAART, pp. 463–474. INSTICC, SciTePress (2020). <https://doi.org/10.5220/0008937004630474>
34. Wang, F., Tian, Y., Wang, X.: A discrete wolf pack algorithm for job shop scheduling problem. In: 2019 5th International Conference on Control, Automation and Robotics (ICCAR), pp. 581–585. IEEE (2019)
35. Wang, L., Cai, J.C., Li, M.: An adaptive multi-population genetic algorithm for job-shop scheduling problem. *Adv. Manuf.* **4**(2), 142–149 (2016)
36. Watanabe, M., Ida, K., Gen, M.: A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Comput. Ind. Eng.* **48**(4), 743–752 (2005)
37. Wegner, P.: A technique for counting ones in a binary computer. *Commun. ACM* **3**(5), 322 (1960). <https://doi.org/10.1145/367236.367286>
38. Wu, Z., Sun, S., Yu, S.: Optimizing makespan and stability risks in job shop scheduling. *Comput. Oper. Res.* **122**, 104963 (2020)
39. Xhafa, F., Abraham, A.: *Metaheuristics for scheduling in industrial and manufacturing applications*, vol. 128. Springer, Heidelberg (2008). <https://doi.org/10.1007/978-3-540-78985-7>
40. Yu, H., Gao, Y., Wang, L., Meng, J.: A hybrid particle swarm optimization algorithm enhanced with nonlinear inertial weight and gaussian mutation for job shop scheduling problems. *Mathematics* **8**(8) (2020). <https://doi.org/10.3390/math8081355>
41. Zang, W., Ren, L., Zhang, W., Liu, X.: A cloud model based DNA genetic algorithm for numerical optimization problems. *Future Gener. Comput. Syst.* **81**, 465–477 (2018)