British Journal of Mathematics & Computer Science

22(3): 1-21, 2017; Article no.BJMCS.33498
ISSN: 2231-0851

Improving Performance of GAs by Use of Selective Breading Evolutionary Process

Farhad Ghassemi-Tari[1*] and Sareh Meshkinfam[2]

[1]Sharif University of Technology, Azadi Ave., P.O.Box 11155-9414, Tehran, Iran.
[2]Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran.

Authors' contributions

This work was carried out in collaboration between both authors. Author FGT designed the study, wrote the protocol, managed the analyses of the study and wrote the first draft of the manuscript. Author SM performed the literature search, developed the algorithms and performed the statistical analysis. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2017/33498
Editor(s):
(1) Mohamed Rabea Eid Said, Department of Science and Mathematics, Assiut University, Egypt.
Reviewers:
(1) Oyeleye Christopher Akinwale, Ladoke Akintola University of Technology, Nigeria.
(2) M. K. Marichelvam, Mepco Schlenk Engineering College, India.
(3) Ankur Singh Bist, KIET Group of Institutions, India.
Complete Peer review History: http://www.sciencedomain.org/review-history/19284

Original Research Article

Received: 18th April 2017
Accepted: 12th May 2017
Published: 1st June 2017

Abstract

In this paper, the use of selective breading evolutionary process for improving the performance of GAs is evaluated. To accomplish this evaluation, the generalized tardiness flow shop scheduling (GTFS) problem is designated. A natural evolutionary GA and two selective breeding Gas are developed for evaluating their performances in solving the proposed problem. An extensive numerical experiment on total of 2250 randomly generated scenarios is conducted to compare the effects of selective breeding mechanism. The effects of the varieties factors on the solution of the algorithms are analyzed by the factorial ANOVA. The computational results reveal that a significant improvement can be obtained if one employs an initial population with better genes.

Keywords: Scheduling; sequencing; natural breading GA; selective breeding GA; generalized tardiness flow shop.

**Corresponding author: E-mail: ghasemi@sharif.edu;*

1 Introduction

Job scheduling is vital tool for manufacturing and engineering and has a major effect on the productivity of a process. Production scheduling incentive is maximization of the efficiency and reduction of the operations costs. Production systems can be classified as batch processing, job shop and flow shop system. A flow shop is a processing system in which the task sequence of each job is fully specified, and all jobs visit the work stations in the same order [1]. The flow shop scheduling measure of performances are generally the time required to finish all jobs or makespan, the average flow time, and the total tardiness of jobs.

This manuscript proposed a set selective breeding genetic algorithm (GA) for the generalized tardiness flowshop (GTFS) problem. The evolutionary process of almost all the cited GAs is inbreeding. In the other hand humans have been domesticating animals for thousand years. In developing countries, the selective breeding has been practiced for the cattle's traits. For producing cattle with better genes, they usually import bulls with better genes and use them for breeding. The whole purpose of selective breeding is to allow individuals with the best sets of genes to reproduce a better next generation [2]. Utilizing the same concept, we investigated the use of selective breeding in GA.

For discussion of the GTFS problem, a model with a due date for completion of each operation of the jobs is considered, while in the traditional tardiness flowshop, there is only a due date for the final operation of each job. However, in most of the real-world projects, the outcomes are delivered through predefined phases and there is an associated due date for each phase. Usually the phases are carried out in a unidirectional precedence structure. Considering n projects to be scheduled, each having m phases, the problem can be modeled as an n jobs-m machines Flow shop scheduling problem with the intermediate jobs due dates. Where the tardiness is defined as the measure scheduling performance, this model is called the generalized version of the tardiness Flow shop scheduling model.

2 Review of Related Works

The GTFS problem was first introduced by Ghassemi-Tari and Olfat [3]. However rare research studies have considered this problem. One of the early work is a study in which four COVERT based heuristic algorithms are proposed for the GTFS [4]. Later several other heuristic algorithms using different rules such as apparent tardiness cost (ATC), slacked-based rule, modified due date (MDD) rules, and other simple rules of SPT and EDD, are also proposed for the problem [5-7]. In spite of the extensive efforts of these works, the use of the metaheuristic type solution approaches have not be considered for the GTFS.

Among the metaheuristics, genetic algorithms are reported as the efficient optimization approaches [8]. Genetic algorithms have been originally proposed by Holland [9]. The use of evolutionary algorithms for shop scheduling problems started around 1980. Two of the first applications of evolutionary algorithms to flow shop scheduling problems have been given by Werner [10]. Another early approach for applying GA to flow shop scheduling problems is the work of Murata et al. [11].

The permutation flow shop scheduling problem (PFSP) has been studied by many researchers However there are little research works considering tardiness flow shop permutation scheduling. Zheng, and Wang [12] investigated the effect of different initialization, crossover and mutation operators on the performances of a genetic algorithm and proposed an effective hybrid heuristic for flow shop scheduling. Later, Wang and Zheng [13] presented a novel and systematical approach based on ordinal optimization and optimal computing budget allocation technique to determine optimal combination of genetic operators for flow shop scheduling problems. Then through a simulation experiment, they have shown that the proposed methodology is able to determine optimal combination of genetic operators and simultaneously to provide a good solution with reasonable performance evaluation for scheduling problem.

Chung et al. [14] developed a genetic algorithm for solving the tardiness permutation flow shop scheduling problems with m-machine, n-job. Use of the GA for solving flow shop problems with the varieties of

objectives are addressed in by Cemil et al. [15]. This study was limited to a flow shop tardiness with a common due date. A GA and a Tabu search algorithm for solving this problem were proposed. Three genetic algorithms for the problem of minimizing total tardiness have been given by Vallada and Ruiz [16].

Onwubulu and Mutingi [17] considered the flow shop problem with three different objectives of minimizing the total tardiness, the number of tardy jobs and a linear combination of both criteria. Babu et al. [18] introduced a genetic algorithm to solve *n*-jobs *m*-machines flow shop scheduling problem to get the optimum results of make-span and total tardiness. Ta et al. [19] considered the *m*-machine tardiness permutation flow-shop scheduling problem and proposed several metaheuristic algorithms. The metaheuristics were compared to a genetic algorithm and concluded the good performances of the metaheuristic algorithms. Other efforts are the works of Rahman [20], Sajadi [21], Shin [22], and Wang et al. [23], Cui et al. [24].

Scheduling with learning effects has received considerable attention recently [25]. Lee and Chung [26] consider a permutation flowshop scheduling problem with learning effects where the objective is to minimize the total tardiness. Other recent studies are the work of Kia et al. [27] and Mou et al. [28].

The review of the related works reveals that the concept of the selective breeding and its prosperities on improving the final solution has not been investigated thoroughly. Also, the use of the metaheuristics for the GTFS problem has been considered only on one study [29]. Moreover, the effects of the dissimilarities of different algorithm routines on the improving the final solution may not be the only source of its improvement, but it may root on the value of the input parameters. This is another deficiency which is realized in the related studies. To overcome these shortcomings, in this research, three GAs are proposed for the GTFS problem. The two of the GAs are developed based on the selective breeding concept. The third GA is proposed based on the natural evolutionary process to evaluate the solution improvement of the selective breeding with respect to the natural breeding. An extensive computational experiment including a factorial-ANVOA is conducted for assessing the solution improvement of the selective breeding algorithms as well as the effects of the verities of input parameters in the solution improvement.

3 The Proposed Solution Algorithms

Due to the combinatorial nature of the mathematical model, we proposed a natural evolutionary and two variants of the selective breeding GA for solving the proposed problem. The concepts of the two variants GA are similar to the natural evolutionary GA except that the initial population of one of the parents is selected from a set of schedules with better genes. The process of developing these algorithms will be described in the following subsections.

3.1 Algorithm 1. The natural evolutionary GA

We proposed the following procedures for generation of the initial population as well as the required operators.

3.1.1 Initial population

For generation of the initial population a chromosome is considered as a permutation schedule of the Flow shop model. Therefore, in a model with *n* jobs and *m* machines there are *m* chromosomes, each having *n* genies which represent the job's sequences. Using pseudo random generation, a uniform density function of $[1, m]$ is employed, and by eliminating the repeated numbers, *m* distinct machines are generated. Then another uniform of $[1, n],$ is used, and again by eliminating the repeated numbers, *n* jobs are assigned to each of the *m* machines randomly. Now for generating a permutation schedules, the same sequence is used for all other machines. By repeating this procedure for every *m* generated machines, we constructed *m* permutation schedules as the initial population.

3.1.2 GA operators

As in general practice a GAs consists of three major operators, namely; selection / reproduction, crossover, and mutation operators. Each of these operators can be describes in the following subsections.

3.1.2.1 Selection/Reproduction operator

The elitist and roulette wheel selection operators are employed for the proposed GA. A probability according to its fittingness values is assigned as a basis for the selection for the further reproduction. In proposed algorithm, the *m* permutation schedules are first sorted by the values of their tardiness in a non-decreasing order. By assigning the rank *k*, from 0 to *m*-1 to the sorted schedule, we defined the function $w(k) = 2(m-k)/m(m+1)$ for assigning a weight to each of the *m* schedules. Now by assigning a corresponding probability according to the defined weights to each population member, and using the density function of $P(x) = \frac{2(m-x)}{m(m+1)}$, we can obtain the cumulative density function (CDF). By generating a random variable *x* from a uniform density function of [0, *m*-1], the selection of each schedule is accomplished according to the inverse of the CDF as bellow:

$$F^{-1}(x) = (2m-1-[(1-2m)^{1/2} - 4m(m-1-(m+1)(1-x))]^{1/2})/2. \quad (1)$$

In the proposed GA, the parents' selection for mutation and crossover operators are also conducted using the roulette wheel selection mechanism. Then all generated chromosomes by crossover and mutation operator are replaced with those having worst objective function values in the current population.

3.1.2.2 Crossover operator

In proposed GA, a crossover rate of $\%p_c$ is used. We let the crossover rate takes different values from 0.70 to 0.90. The Modified Order Crossover (MOX) is employed as the crossover operator. In MOX operator by a random number generation mechanism a cut point is determined. Then the genes in the left segment of the cut from first (second) parent are copied to the genes in the left segment of the cut in the first (second) offspring. The genes from second (first) parent, after omitting the selected genes are designated and mapped to right of the cut of the first (second) offspring in the same order. Through this procedure two offspring are generated and kept in a file to be considered after the mutation process. Fig. 1 presents an illustrative example of the crossover operator.

3.1.2.3 Mutation operator

In the proposed algorithm, the mutation is applied after $p_m = 100*(1-p_c)$ percent, in which p_c is the percentage of crossover iterations. The mutation is performed by the one point inversion method. First, we generated a random variable from a uniform density function of [0, *n*]. The value of this random variable specifies the cut point in the string of the genes. Then the genes in the left segment of the cut of the parent are copied to the left segment of the cut of the genes of the offspring in the same order. The remaining genes of the parent are then reversed and mapped to right of the cut of the offspring. Fig. 2 presents an illustrative example of the mutation operator.

It is to be noted that, we select 20 as the number of iterations. This is due to the fact that in the process of conducting the computational experiment, we performed an extensive analysis on different factors, such as number of jobs, number of machines, crossover rate, and different factor levels. The computation process failed for the most of the higher factors levels where the iteration numbers are set to the greater than 20. Therefore, in order to have a consistence results for all different factor levels, the number of iterations is set to 20.

3.1.3 Steps of the GA

The steps of the proposed GA can be summarized by the following algorithmic procedures.

Step1. Generate the initial population and calculate the tardiness of the parents.

Step2. Let $p_c = 0.70, \text{and} It = 1.$

Step3. Perform crossover iteration, and calculate the tardiness of the offspring to save it in File *OFC*.

Step4. Perform Mutation iteration, and calculate the tardiness of the offspring to save it in File *OFM*.

Step5. Select offspring with the rate of $100p_c$ from *OFC* and offspring with the rate of $100(1-p_c)$ percent from *OFM*, and select the least *m* tardy schedules.

Step6. If $It = 20,$ go to Step7, otherwise let $It = It + 1,$ and go to Step3.

Step7. Designate the sequence with the smallest tardiness as the final solution for this p_c. If $p_c = 0.90,$ stop. Otherwise let $p_c = p_c + 0.05,$ and go to Step 3.

3.2 Selective breeding algorithms

Three different sequencing rules of shortest processing time (SPT), earliest due date (EDD), and combined SPT-EDD, are employed for generation of the initially selected parent and two algorithms each with three variants of one of the initial parent are proposed. Before illustrating the outlines of these algorithms let us first describe the generation of the selected initial population.

3.2.1 Generation of the selected initial parent

To generate the selected initial population, we employed some heuristic sequencing rules, such as SPT, EDD, and combined SPT-EDD. Since in the generalized tardiness flow shop problem a due date is defined for every operation of the jobs, it would be worthwhile to deal each of the machines distinctly as a single machine tardiness problem for developing some simple sequencing rules. Let us first propose the following definitions and theorems.

Definition 3.2.1.1: *Let S represents a schedule in which job i proceeds job j and S' represents another schedule which is identical to S except that job j proceeds job i.*

Definition 3.2.1.2: *Let the total tardiness of job i and j in schedule S and schedule S' are denoted by* T_{ij}, *and* T_{ji} *respectively. By letting B(i) and B(j) as the available time of job i and j to be processed respectively, then the value of* T_{ij}, *and* T_{ji} *can be determined as follows:*

$$T_{ij} = T_i(S) + T_j(S) = \max\{B(i) + t_i - d_i, 0\} + \max\{B(j) + t_j - d_j, 0\}. \quad (2)$$

$$T_{ji} = T_j(S') + T_i(S') = \max\{B(j) + t_j - d_j, 0\} + \max\{B(i) + t_i - d_i, 0\}. \quad (3)$$

Since the rest of jobs in both schedules have the same sequence, only different values of T_{ij}, *and* T_{ji} *alter the total tardiness of the both schedules.*

Based on the above definitions the following theorems are proposed:

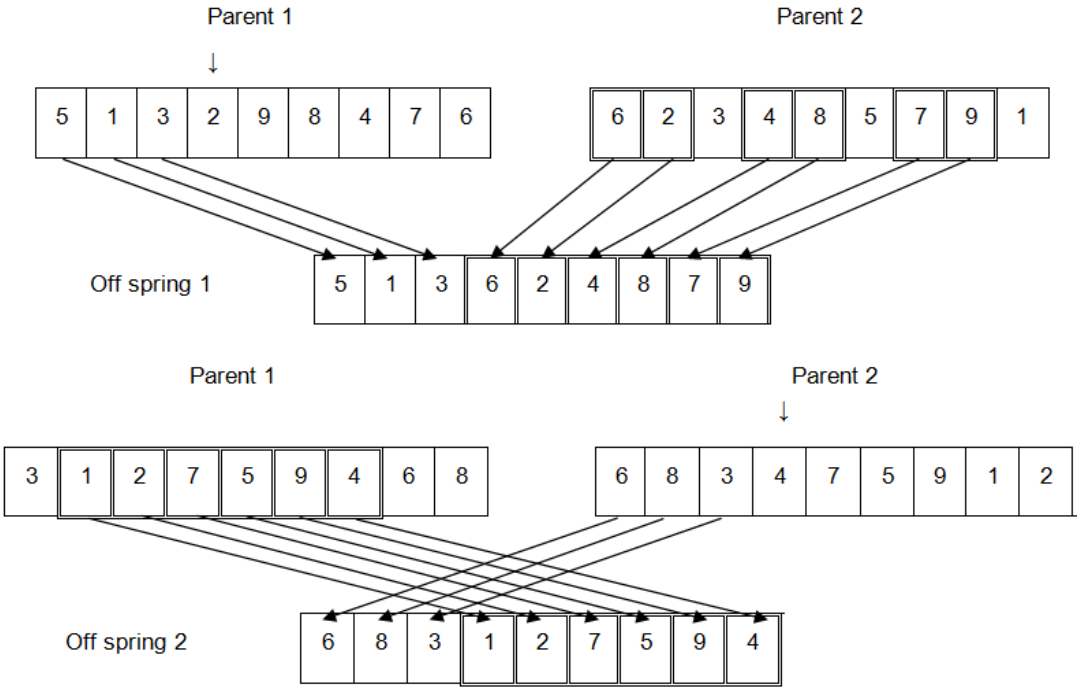

Fig. 1. An example of the crossover operator

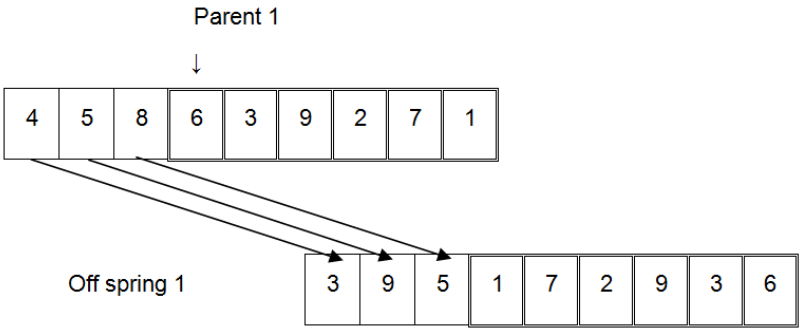

Fig. 2. An example of the mutation operator

Theorem 3.2.1. *Assuming* $t_i \geq t_j$ and $d_i \geq d_j$, *then SPT provides a schedule, which is better than or equal to the non SPT schedule with the objective of minimizing the total tardiness of the jobs.*

Proof. Based on the values of the total tardiness of Schedule S and Schedule S' which are determined by the following equation, it can be verified that $T_{ij} > T_{ji}$.

$$T_{ij} = \max\{B(i) + t_i + t_j - d_j, 0\}. \tag{4}$$

$$T_{ji} = \max\{B(j) + t_j - d_j, 0\} + \max\{B(i) + t_i + t_j - d_i, 0\} \tag{5}$$

If $B(i)+t_i \le d_i$, *it can be deducted that* $T_{ij} > T_{ji}$. *Therefore, it is preferable to have job j with smaller processing time (or at most equal processing time to job i) precede job i. By using the same reasoning for every other two jobs we can conclude that the SPT schedule is superior to the non SPT schedule. Now if* $B(i)+t_i > d_i$. *Then,*

$$T_{ij} = B(j)+t_i-d_i+B(j)+t_i+t_j-d_j. \tag{6}$$

$$T_{ji} = \max\{B(j)+t_j-d_j,0\}+B(i)+t_i+t_j-d_i. \tag{7}$$

We then have $T_{ij}-T_{ji} = B(i)+t_i-d_j-\max\{B(j)+t_j-d_j,0\}$. *If the maximum in the last term is zero, then the condition implies that* $T_{ij} \ge T_{ji}$; *and if the maximum in the last term is positive, then*

$T_{ij}-T_{ji} = B(i)+t_i-d_j-B(j)+t_j-d_j = t_i-t_j \ge 0$. *Therefore, it can be concluded that* $T_{ij} \ge T_{ji}$, *so it is preferable to have job j with smaller processing time (or at most equal processing time to job i) precede job i. By using the same reasoning for every other two jobs we can conclude that the SPT schedule is superior to the non SPT schedule.*

Theorem 3.2.2. *Assuming* $t_i \ge t_j$, and $d_i < d_j$, *then EDD provides a schedule, which is better than or equal to the non EDD schedule with the objective of minimizing the total tardiness of the jobs unless for the case that* $B(j)+t_j > d_j$, *in which SPT provides a better schedule.*

Proof.

$$T_{ij} = B(i)+t_i-d_i. \tag{8}$$

$$T_{ji} = B(j)+t_j+t_i-d_i. \tag{9}$$

If $B(j)+t_i > d_i$, *and* $B(i)+t_i+t_j \le d_j$. *then* $T_{ji} \ge T_{ij}$. *So it is preferable to have job i (the job with the earlier due date) precede job j. Now if* $B(i)+t_i > d_i$. *and* $B(j)+t_j \le d_j < B(i)+t_i+t_j$. *Then we have,*

$$T_{ij} = B(i)+t_i-d_i+B(j)+t_i+t_j-d_j. \tag{10}$$

$$T_{ji} = B(j)+t_j+t_i-d_i \tag{11}$$

Then $T_{ij}-T_{ji} = B(i)+t_i-d_j$. *Therefore, we can conclude that, it is preferable to have job i (the job with the earlier due date) precede job j unless* $B(i)+t_i > d_j$, *in which case job j (the shorter job) may precede job i. This due to the fact that, if* $B(j)+t_j > d_j$, *then we will have:*

$$T_{ij} = B(i)+t_i-d_i+B(j)+t_i+t_j-d_j \tag{12}$$

$$T_{ji} = B(j)+t_j-d_j+B(i)+t_j+t_i-d_i. \tag{13}$$

Then $T_{ij} - T_{ji} = t_i - t_j \geq 0$, *and Therefore, Case 2.2.3 yields* $T_{ij} \geq T_{ji}$, *so it is preferable to have job j (the shorter job) precede job i. Again by using the same reasoning for every other two jobs the proof is completed.*

If $B(i) + t_i \leq d_i$, *based on the values of the total tardiness of Schedule S and Schedule S' which are determined by the following equation, it can be verified that it can be deduct that* $T_{ij} > T_{ji}$.

$$T_{ij} = \max\{B(i) + t_i + t_j - d_j, 0\}. \tag{14}$$

$$T_{ji} = \max\{B(j) + t_j - d_j, 0\} + \max\{B(i) + t_i + t_j - d_j, 0\}. \tag{15}$$

Therefore, it is preferable to have job j with smaller processing time (or at most equal processing time to job i) precede job i. By using the same reasoning for every other two jobs we can conclude that the SPT schedule is superior to the non SPT schedule.

Now if $B(i) + t_i > d_i$. *Then*

$$T_{ij} = B(i) + t_i - d_i + B(j) + t_i + t_j - d_j. \tag{16}$$

$$T_{ji} = \max\{B(j) + t_j - d_j, 0\} + B(i) + t_i + t_j - d_i. \tag{17}$$

We then have $T_{ij} - T_{ji} = B(i) + t_i - d_j - \max\{B(j) + t_j - d_j, 0\}$. *If the maximum in the last term is zero, then the condition implies that* $T_{ij} \geq T_{ji}$; *and if the maximum in the last term is positive, then* $T_{ij} - T_{ji} = B(i) + t_i - d_j - B(j) + t_j - d_j = t_i - t_j \geq 0$. *Therefore, it can be concluded that* $T_{ij} \geq T_{ji}$, *so it is preferable to have job j with smaller processing time (or at most equal processing time to job i) precede job i. By using the same reasoning for every other two jobs we can conclude that the SPT schedule is superior to the non SPT schedule.*

Based on these theorems if one considers each machine of the flow shop model distinctively, SPT and EDD sequencing rules can provide a near optimal tardiness of each machine. This could be a reasonable assumption due to the fact that there is a due date for every operation of the jobs in the permutation schedule. Now in obtaining the permutation schedules, if we apply these rules to all machines of the generalized flow shop we obtain *m* distinct permutation schedules and from those we can generate the initial population of one of the parents. To distinguish the job number and its position in the sequence, it would be convenient to use brackets to indicate position of job in the sequence. Using this concept, [*k*] = *l* means that the k^{th} job in the sequence is job *l*. Similarly, $t_{[k]l}$ refers to the processing time of the k^{th} job in sequence being processed on machine *l*. Based on the above sequencing rules three subroutines are proposed for generating one of the initial populations. These subroutines are presented as the followings:

Subroutine 1. SPT schedules

Step 1. Let *k*=1.
Step 2. Schedule jobs in non-decreasing order of t_{ik}; i.e. $t_{[1]k} \leq t_{[2]k} \leq \cdots \leq t_{[n]k}$.
Step 3. Use this sequence for all the *m* machines and calculate the total tardiness of this permutation schedule.
Step 4. If *k*=*m*, go to step 5, otherwise let *k*=*k*+1, and, then go to step 2.
Step 5. Select the generated *m* schedules for one of the selective parent, and stop.

Subroutine 2. EDD schedules

Step 1. Let k=1.
Step 2. Schedule jobs in non-decreasing order of d_{ik}; i.e. $d_{[1]k} \le d_{[2]k} \le \cdots \le d_{[n]k}$.
Step 3. Use this sequence for all the m machines and calculate the total tardiness of this permutation schedule.
Step 4. If k=m, go to step 5, otherwise let k=k+1, and, then go to step 2.
Step 5. Select the generated m schedules for one the selective parent, and stop.

Subroutine 3. Combined SPT-EDD schedules

Step 1. Let k=1.
Step 2. Schedule jobs in non-decreasing order of t_{ik}; i.e. $t_{[1]k} \le t_{[2]k} \le \cdots \le t_{[n]k}$.
Step 3. Use this sequence for all the m machines and calculate the total tardiness of this permutation schedule.
Step 4. Schedule jobs in non-decreasing order of d_{ik}; i.e. $d_{[1]k} \le d_{[2]k} \le \cdots \le d_{[n]k}$.
Step5. Use this sequence for all the m machines and calculate the total tardiness of this permutation schedule.
Step 6. If k=m, go to step 7, otherwise let k=k+1, and, then go to step 2.
Step 7. Select m schedules with the lowest tardiness from the generated 2$m$schedules and designate them as one of the selective parent, and then stop.

As it is mentioned earlier, evaluation of the effect of the selected breading on the solution of the GA is the major purpose of this study. To achieve this goal, two other GA algorithms, in addition to the natural evolutionary GA (Algorithm1), are proposed. In Algorithm 2 and Algorithm 3, contrary to the customary random selection, one of the initial parents is nominated by the best schedule, presumably as parent with the better genes. In Algorithm 2, the selected initial parent remains unchanged throughout the all evolutionary process of the generating new populations. In Algorithm 3, we let that the initially selected parent to be evolved throughout the evolutionary genetic process. A more detail description of the algorithms will be illustrated in the following subsections.

3.2.2 Algorithm 2-Selective breeding algorithm (unchanged initially selected parent)

Algorithm 2 differs from the proposed natural evolutionary GA (Algorithm 1) by the generation of the population of one of the parents. In Algorithm 1 both parents are generated randomly and the next populations are generated through the use of the crossover and the mutation process. However, in Algorithm 2 we let one of the parents is selected from the population with better genes. For a better evaluation, three versions of GA are considered for the initial selecting population with better genes. Bellow, designates these versions:

Version 1. GA with the SPT schedules.
Version 2. GA with the EDD schedules.
Version 3. GA with the best of SPT and EDD schedules.

To begin with, the above three subroutines are first executed and among their solutions, the best schedule with the minimum tardiness is nominated as the initially selected parent for each version. Using the selected schedules as one of the parents, the other procedures of Algorithm 2 are similar to Algorithm 1. The following summarizes the steps of Algorithm 2.

Step1. Let k=1
Step 2. Generate the initial selected parent using version k.
Step3. Generate the other initial parent randomly. Let $p_c = 0.70, \text{and } It = 1.$

Step4. Perform crossover iteration and calculate the tardiness to save it in File *OFC*.
Step5. Perform Mutation iteration and calculate the tardiness to save it in File *OFM*.
Step6. Select offspring with the rate of $100p_c$ from *OFC* and offspring with the rate of $100(1-p_c)$ percent from *OFM*, and select the least *m* tardy schedules as one of the new parents, and the initially selected parent as the other new parent.
Step7. If $It = 20$, go to Step 8, otherwise let $It = It + 1$, and go to Step 4.
Step8. Designate the sequence with the smallest tardiness as the final solution for this p_c. If $p_c = 0.90$, go to Step 9, otherwise let $p_c = p_c + 0.05$, and go to Step 4.
Step 9. If *k*=3 stop, otherwise let *k*=*k*+1, go to Step 2.

3.2.3 Algorithm 3-Selective breeding algorithm (evolving initially selected parent)

Similar to Algorithm 2, three versions of GA are considered for the initial selecting population with better genes. Algorithm 3 differs from Algorithm 2 by letting both of the initially selected parents originates form selecting populations with better genes. Other than this the rest of algorithmic procedures of Algorithm 3 are similar to Algorithm 2. The following summarizes the steps of Algorithm 3.

Step1. Let *k*=1
Step 2. Generate the initial selected parent using version *k*.
Step3. Chose the two initial parents by the selection procedure (described in Section *4.1.2.1)*. Let $p_c = 0.70, \text{and } It = 1.$
Step4. Perform crossover iteration and calculate the tardiness to save it in File *OFC*.
Step5. Perform Mutation iteration and calculate the tardiness to save it in File *OFM*.
Step6. Select offspring with the rate of $100p_c$ from *OFC* and offspring with the rate of $100(1-p_c)$ percent from *OFM*, and select the least *m* tardy schedules.
Step7. If $It = 20$, go to Step 8, otherwise let $It = It + 1$, and go to Step4.
Step8. Designate the sequence with the smallest tardiness as the final solution for this p_c. If $p_c = 0.90$, go to Step 9, otherwise let $p_c = p_c + 0.05$, and go to Step 4.
Step 9. If *k*=3 stop, otherwise let *k*=*k*+1, go to Step 2.

4 Computational Experiments

An extensive computational experiment, based on factorial analysis of variance (ANOV) is conducted for evaluating the effect of different factors on the improvement of the initial solution for different versions of the proposed algorithms. The improvement is determined by the difference of the final solution and the initial solution by the fowling relation:

$$\%I = (IT - FT) * 100 / IT. \tag{18}$$

Where %*I*, designates the percentage improvement in the schedule tardiness obtained by applying each individual GA, *IT* represents the initial tardiness of the schedule at the begging of the corresponding GA, and *FT* indicates tardiness of the final schedule obtained after termination of the corresponding GA.

An extensive computational experiment consisting of 2250 randomly generated scenarios are performed for evaluating the effects of different factors at different level to test the hypothesis of the equality of the improvement obtained by the different algorithms. More precisely a factorial ANOVA is conducted for algorithm 1, in which the number of job, the number of machine, and the crossover rate are designated as the independent variables and the percent improvement is designated as the dependent variable. Also, a three

factorial ANOVA is performed for evaluating the effect of three factors, such as the number of job, the number of machines and the type of the initially selected schedule (STP, EDD, or combine STP-EDD) as one of the parents on the equality of the percentage of improvement (%I) obtained by Algorithm 2 and Algorithm 3. To perform the ANOVA, five test problems are randomly generated for each instance. Then, the proposed algorithms are coded by C++ and run on a Core i5 PC with 8 GB Ram, using SAS software. The following subsections describe the generation of the test problems and the results of these experiments respectively.

4.1 Generation of the test problems

For generating an unbiased set of the test problems, the concept of pseudo random generation is employed. The test problems are randomly generated with various sizes, in terms of both the number of jobs and the number of machines and are respectively classified according to the values of m = 3, 5, 7, 9, 10, and n= 15, 25, 50, 100, 200, 400. For each combination of job and machine, five instances are randomly generated and solved by 20 iterations with the crossover rates (*CR*s) of 0.70, 0.75, 0.80, 0.85, and 0.90. It is to be noted that, several computational experiments are also conducted on the different factor levels with the number of iteration greater than 20. However due to computational complexity of the higher factor level values, most of them are failed. Therefore, in order to have a consistence results for all different factor levels, the number of iterations is set to 20.

For the number of jobs, the following sizes are considered: 8, 10, 12, 15, 17, 20, 25, 30, 40, 50, 75, 100, 200, 300, 400 and 500. For the machines, we considered problems with 5, 10 and 20 machines. For each job, the processing times on the various machines are generated from a uniform distribution over the integers 1 to 100, while an integer weight is obtained from a uniform distribution [1,10]. Finally, for each job, an integer due date was generated from the uniform distribution. Both the tardiness factor and the range of due dates parameters were set at 0.2, 0.4, 0.6, 0.8 and 1.0. For each combination of n and m, 50 instances were randomly generated. As the result, a total of 1250 instances were generated for each problem size.

For each job, the processing times on the various machines, an integer value is generated from a normal distribution with $\mu = 4$ and $\delta = 5$. Finally, for each job, an integer due date was generated from the normal distribution with $\mu = 9$ and $\delta = 4$.

4.2 Computational results

The tables illustrating the results of ANOV are presented in the Appendix 1. In this appendix, a class of experiments are devoted to the hypothesis on the equality of %*I* with respect to the effect of different factors. Table A1-1, in the appendix, presents the result of ANOVA on effect of the three factors on the percentage improvement (%*I*) for Algorithm 1. The three factors A, B and C represent the number of jobs (*n*), the number of machines (*m*), and the different crossover rates (*CR*), respectively. This table reveals that the equality of the percentage improvement according to the different levels of the three factors and their interactions is rejected with the probability of greater than .9999 (1-.0001). Considering a distinct crossover rate, a similar analysis is conducted on Algorithm 1 and algorithm 2. For each individual crossover rate, a 3-factorial analysis conducted in which factor A, B, and C respectively designates the number of jobs, the number of machines, and the different initial selected parent (*ISP*) with the levels of SPT, EDD, and combined SPT-EDD. The results are illustrated in Table A1-2 through A1-7 of Appendix 1. Based on the content of these tables, it can be concluded that the hypothesis of the equality of the percentage improvement is rejected for all of the individual design with probability of greater than .9999 (1-.0001).

It would be worthy, to illustrate how the percent improvement obtained by the different algorithms varies according to different input parameters. To reveal this variations Table 1 and Table 2 are provided. Table 1 presents the percent improvement of Algorithm 1 with respect to the variations of the crossover rates and the number of machines. The efficiency of this algorithm is ranked for the variation of the crossover rates and the number of jobs, based on the values of the percent improvement. In this table, there is a consistency on the efficiency of Algorithm1 and the crossover rates. It can be realized that the crossover rates of .70 has the first ranking and as the crossover rate increases the efficiency decreases uniformly. Similarly, as the number of machines increase the efficiency virtually increases.

Table 2 provides the percent improvement of Algorithm 1 with respect to the variations of the crossover rates and the number of jobs. Likewise, the efficiency of this algorithm decrease as the crossover rate increases uniformly. However, as the number of jobs increase the efficiency decrease consistently.

To illustrate the percent improvement on the solution of Algorithm 2 according to the number of machines and the number of jobs Table 3, and Table 4, is constructed respectively. Table 3 illustrates the percent improvement obtained by Algorithms 2 according to the variations of the initially selected parent and the number machines. The percent improvement is also ranked according to the number of machines and the type of the initially selected parent. It can be realized selection of the initial parent by the STP rule seems to have a better efficiency and as the number of machines increase the efficiency virtually increases.

Table 4 presents the percent improvement obtained by Algorithms 2 according to the variations of the initially selected parent and the number jobs. The percent improvement is also ranked according to the number of jobs and the type of the initially selected parent. In this case the selection of the initial parent by the STP-EDD rule appears to have a better efficiency and as the number of jobs increase the efficiency consistently decrease.

Similarly, to illustrate the percent improvement on the solution of Algorithm 3 according to the number of machines and the number of jobs Table 5, and Table 6, is created. Table 5 demonstrates the percent improvement of on the solution of Algorithm 3, for different initially selected parent, according to the number of machines. The percent improvement is also ranked according to the number of machines and the type of the initially selected parent. It can be realized that the performance of the percent improvement on the solution of Algorithm 3 executes very similar to the performance of Algorithm 2. More clearly, the selection of the initial parent by the STP rule seems to have a better efficiency and as the number of machines increase the efficiency virtually increases.

Table 1. Percent improvement of Algorithm 1 and ranking per the number of machines and the *CR*s

m	***CR***					**Total**	**Ranking**
	0.70	**0.75**	**0.80**	**0.85**	**0.90**		
3	6.77	6.29	6.24	6.43	6.14	6.37	*5*
5	9.00	8.56	8.50	8.52	7.93	8.50	*4*
7	11.32	9.63	9.97	9.63	9.34	9.98	*3*
9	9.95	11.44	9.94	10.25	10.71	10.46	*1*
10	11.24	9.79	9.85	10.23	9.85	10.19	*2*
Average	8.16	7.75	7.55	7.65	7.48	9.10	
Ranking	*1*	*2*	*4*	*3*	*5*		

Table 2. Percent improvement of algorithm 1 and ranking per the number of machines and the *CR*s

n	***CR***					**Total**	**Ranking**
	0.70	**0.75**	**0.80**	**0.85**	**0.90**		
15	17.05	16.28	16.55	17.41	16.50	16.76	*1*
25	13.21	14.37	13.08	13.41	12.80	13.37	*2*
50	9.41	9.54	9.56	8.82	8.72	9.21	*3*
100	6.45	6.27	6.03	6.28	6.56	6.32	*4*
200	6.41	4.84	4.60	4.73	4.62	5.04	*5*
400	5.42	3.57	3.57	3.42	3.56	3.91	*6*
Ranking	*1*	*2*	*4*	*3*	*5*		

Table 6 describes the percent improvement of on the solution of Algorithm 3, for different initially selected parent, according to the number of jobs. The percent improvement is also ranked according to the number of machines and the type of the initially selected parent. It can be realized that the performance of the percent improvement on the solution of Algorithm 3 according to the number of jobs performs very similar to the

performance of Algorithm 2. More clearly, as the number of jobs increase the efficiency virtually decreases. However, the selection of the initial parent by the STP-EDD rule seems to have a better efficiency.

Table 3. Percent improvement and ranking of Algorithm 2 per initial parent and *m*

m	**Initial selected parent**			**Total**	**Ranking**
	SPT	**EDD**	**SPT-EDD**		
3	7.70	6.88	7.66	7.42	*5*
5	9.20	8.86	9.45	9.17	*4*
7	9.92	10.03	10.34	10.10	*3*
9	9.59	10.13	10.73	10.15	*1*
10	9.49	9.98	10.58	10.01	*2*
Ranking	*1*	*3*	*2*		

Table 4. Percent improvement and ranking of Algorithm 2 per the initial parent and *n*

n	**Initial selected parent**			**Total**	**Ranking**
	SPT	**EDD**	**SPT-EDD**		
15	18.11	17.12	18.83	18.02	*1*
25	13.33	13.73	14.42	13.82	*2*
50	9.68	9.89	9.65	9.74	*3*
100	5.89	6.22	6.73	6.28	*4*
200	4.44	4.58	4.77	4.60	*5*
400	3.63	3.52	4.11	3.75	*6*
Ranking	*2*	*3*	*1*		

Table 5. Percent improvement and ranking of Algorithm 3 according to the initial parent and the number of machines

m	**Initial selected parent**			**Total**	**Ranking**
	SPT	**EDD**	**SPT-EDD**		
3	13.31	5.75	4.74	7.93	*5*
5	14.05	7.92	6.83	9.60	*4*
7	13.91	9.66	7.80	10.46	*3*
9	14.89	10.86	9.18	11.64	*1*
10	15.32	11.81	8.63	11.92	*2*
12	14.29	9.20	7.44	10.31	
Ranking	*1*	*2*	*3*		

Table 6. Percent improvement and ranking of Algorithm 3 according to the initial parent and the number of jobs

n	**Initial selected parent**			**Total**	**Ranking**
	SPT	**EDD**	**SPT-EDD**		
15	18.63	16.20	11.95	15.59	*1*
25	17.61	13.71	10.17	13.83	*2*
50	14.33	9.60	8.50	10.81	*3*
100	13.15	6.79	6.10	8.68	*4*
200	11.36	5.60	4.56	7.17	*5*
400	10.69	3.31	3.34	5.78	*6*
500	14.29	9.20	7.44	10.31	
Ranking	*1*	*2*	*3*		

Finally, to conclude the computational experiment Table 7 is provided. In this table, the percent improvements selective breeding algorithms (Algorithm 2 and Algorithm 3) with respect to the natural breeding algorithm (Algorithm 1), for different values of *n*, *m*, and *CR* are determined and reported. Based on the content of this table, it can be globally concluded that the performance of the algorithm 3 is greatly better than Algorithm2. Another notable inference is the growth on the percent improvement as the result of the increase on the value of *CR*. This result somehow would be predictable due to the structure of the proposed GA. In the proposed GA, We let a mutation to be is performed for every other iteration. Since the mutation may reverse the progress of optimization, the higher *CR* and therefore the lower mutation rate, (1-*CR*), would be preferable.

Table 7. Percent improvement of the Algorithm 2 and 3 with respect to Algorithm 1 for different CR

n	***m***	**% 70.00**		**% 75.00**		**% 80.00**		**% 85.00**		**% 90.00**	
		Alg. 2	**Alg. 3**	**Alg. 2**	**Alg. 3**	**Alg. 2**	**Alg. 3**	**Alg. 2**	**Alg. 3**	**Alg. 2**	**Alg. 3**
15		5.81	10.98	13.66	15.85	10.83	16.77	12.10	17.47	8.70	14.37
25		13.65	18.56	18.57	20.54	22.83	23.64	24.79	28.79	31.69	31.40
50		18.13	26.51	66.39	29.30	5.95	17.73	24.24	29.77	27.45	33.33
100	3	15.07	24.68	23.55	31.06	20.86	30.46	10.14	20.65	8.49	17.26
200		1.12	5.17	10.60	11.93	14.00	19.18	1.02	10.32	25.80	23.01
400		24.02	34.14	33.20	38.03	25.07	31.18	20.11	21.98	27.73	33.64
15		13.19	17.07	15.04	20.80	16.09	21.91	13.79	19.12	19.11	23.11
25		15.06	14.40	0.00	6.82	11.65	11.02	12.99	14.56	10.16	12.36
50		3.82	10.21	8.42	14.52	8.72	15.49	8.35	13.83	23.69	27.32
100	5	-4.56	8.43	4.83	17.74	4.55	4.84	4.17	11.01	3.08	9.93
200		-5.68	3.10	3.24	4.77	5.46	7.41	9.57	11.95	10.22	6.17
400		12.38	9.47	12.90	7.92	9.89	7.67	17.53	8.43	18.57	8.03
15		11.65	9.62	17.29	15.63	7.31	6.27	9.56	11.37	15.12	18.32
25		14.08	17.29	5.56	5.57	9.12	8.90	0.74	2.41	6.64	9.93
50		4.47	6.56	14.74	16.41	3.47	7.59	4.91	12.04	12.42	17.71
100	7	-3.62	-4.00	-0.36	-4.46	6.44	3.77	5.80	2.06	7.31	2.39
200		-3.29	3.21	4.76	1.72	1.13	8.23	-7.89	-2.85	2.52	1.89
400		-6.00	-6.94	16.67	15.27	2.16	-3.05	14.04	11.40	25.13	7.48
15		14.48	19.48	6.19	15.38	9.86	15.71	5.41	11.48	1.69	6.60
25		3.57	14.16	0.86	13.48	1.55	14.61	0.59	10.86	-3.36	7.23
50		7.49	15.78	18.16	42.43	2.94	8.39	10.37	16.37	9.79	16.70
100	9	50.53	2.77	-3.67	14.26	0.00	15.68	0.73	8.19	-4.80	6.17
200		-3.91	52.51	-3.10	-3.91	8.73	2.79	4.79	4.11	-4.08	5.04
400		6.50	14.77	10.19	19.02	20.39	21.77	15.55	20.75	8.35	11.20
15		5.18	10.19	19.29	23.11	15.77	20.15	2.33	10.59	12.36	17.11
25		21.02	22.05	0.60	8.26	-7.25	-1.13	-0.87	4.90	-5.21	6.13
50		0.46	12.32	-0.21	4.51	2.77	9.15	4.96	10.87	9.59	11.91
100	10	11.03	10.73	10.84	14.03	1.02	10.30	0.80	0.34	-2.02	0.17
200		-0.12	4.46	-1.14	-4.44	-0.26	2.23	8.12	8.28	3.65	7.53
400		12.39	25.78	-12.77	10.17	6.49	23.05	16.48	25.73	7.58	24.31
Average percent Improvement		7.17	13.78	6.34	14.19	6.80	12.72	6.72	12.56	8.91	13.92

Additional experiment is conducted to evaluate the deviation of the average solutions of Algorithm 1, Algorithm 2, and Algorithm 3 from the optimal solution. To conduct this experiment, 24 additional test problems are randomly generated and classified per the number of jobs and the number of machines. Using the crossover rate of %90, the test problems are solved by the proposed algorithm and CPLX. V12. The

results of this experiment are depicted in Table 8. In this table, the percentage of deviations of the solutions of Algorithm 1, Algorithm 2, and Algorithm 3 from the optimal solution are determined by the following relation:

$$\%\text{Deveation} = \frac{\text{the proposed algorithm solution} - \text{the optimal solution}}{\text{the proposed algorithm solution}}$$

The result revealed that the deviation of the solution of Algorithm 3 from the optimal solution, in average, is % 1.5 which is considerably smaller than the other two algorithms. This indicates that the Algorithm 3 with the evolving initially selected parent GA performs better than the natural evolving GA (Algorithm 1) as well as the selective breeding algorithm in which the initially selected parent remains unchanged throughout the algorithm iteration's (Algorithm2).

To evaluate the performance of the proposed approach in comparing to the existing works, the only available work considering GA for the GTFS problem is selected from the cited literature [], Then the cited algorithm (CA) is compared with Algorithm (3) and the results are summarized in Table 9.

Table 8. Deviation of Algorithm1, Algorithm 2, and Algorithm 3 from the optimal solution

Problem no.	***m***	***n***	**Solution of Alg. 1**	**Solution of Alg. 2**	**Solution of Alg. 3**	**The optimal solution**	**Percent deviation of Alg. 1 from the optimal**	**Percent deviation of Alg. 1 from the optimal**	**Percent deviation of Alg. 1 from the optimal**
1		5	17.9	17.2	17.1	17.1	4.47	0.58	0.00
2		6	29.7	28.6	27.9	27.6	7.07	3.50	1.09
3	5	7	42.6	43	42.1	40.7	4.46	5.35	3.44
4		9	48.1	47.5	46.3	46.1	4.16	2.95	0.43
5		10	49.9	49.9	49.1	48.7	2.40	2.40	0.82
6		12	74.1	75.2	69.8	69.1	6.75	8.11	1.01
7		5	23.1	23	22.2	21.3	7.79	7.39	4.23
8		6	30.4	30.3	29.5	28.8	5.26	4.95	2.43
9	7	7	49.8	48.8	47.9	47.5	4.62	2.66	0.84
10		9	68.5	65.1	64.4	63.7	7.01	2.15	1.10
11		10	87.1	86.2	83.2	82.7	5.05	4.06	0.60
12		12	112.2	109.7	109.7	-	-	-	-
13		5	52.1	51.1	50.4	50.1	3.84	1.96	0.60
14		6	69.7	69.6	69.2	68.5	1.72	1.58	1.02
15	9	7	89.5	89.1	88.2	87.6	2.12	1.68	0.68
16		9	100.1	99.8	99.3	95.1	5.00	4.71	4.42
17		10	115.9	116.7	109.6	-	-	-	-
18		12	172.3	165.8	164.9	-	-	-	-
19		5	55.8	53.6	52.3	51.2	8.24	4.48	2.15
20		6	76.8	73.6	70.7	70	8.85	4.89	1.00
21	10	7	99.6	98.9	98.1	96.5	3.11	2.43	1.66
22		9	104.5	101.7	100.1	98.2	6.03	3.44	1.93
23		10	119.7	119.9	117.4	-	-	-	-
24		12	134.6	132.3	130.8	-	-	-	-
Average							**5.16**	**3.65**	**1.55**

Table 9. Comparing the solution of Algorithm 3 with the selected algorithm (CA) from the literature

Problem no.	*m*	*n*	Solution of CA	Solution of Alg. 3	The optimal solution	Percent deviation of CA from the optimal	Percent deviation of Alg.3 from the optimal	Percent deviation of CA from alg.3
1		5	17.4	17.1	17.1	1.75	0	1.75
2		6	27.9	27.9	27.6	1.09	1.09	0
3	5	7	42.9	42.1	40.7	5.41	3.44	1.90
4		9	47.7	46.3	46.1	3.47	0.43	3.02
5		10	51.6	49.1	48.7	5.95	0.82	5.09
6		12	70.4	69.8	69.1	1.88	1.01	0.86
7		5	23.4	22.2	21.3	9.86	4.23	5.41
8		6	31.4	29.5	28.8	9.03	2.43	6.44
9	7	7	48.7	47.9	47.5	2.53	0.84	1.67
10		9	67.5	64.4	63.7	5.97	1.10	4.81
11		10	88.1	83.2	82.7	6.53	0.60	5.89
12		12	114.4	109.7	-	-	-	4.28
13		5	52.7	50.4	50.1	5.19	0.60	4.56
14		6	73.1	69.2	68.5	6.72	1.02	5.64
15	9	7	92.3	88.2	87.6	5.37	0.68	4.65
16		9	101.1	99.3	95.1	6.31	4.42	1.81
17		10	113.6	109.6	-	-	-	3.65
18		12	171.6	164.9	-	-	-	4.06
19		5	57.1	52.3	51.2	11.52	2.15	9.18
20		6	74.7	70.7	70	6.71	1.00	5.66
21	10	7	101.5	98.1	96.5	5.18	1.66	3.47
22		9	103.9	100.1	98.2	5.80	1.93	3.80
23		10	121.3	117.4	-	-	-	3.32
24		12	137.1	130.8	-	-	-	4.82

5 Conclusions

In present work, an attempt has been made to conduct a thorough evaluation of the effect of the selective breeding concept on the solution of GAs. To assess this effect a GTFP problem is selected as a nominee and two selective breeding are proposed for the problem. The SPT, EDD and STP-EDD rules are employed to generate a set of schedules which construct one of the initial populations of the proposed selective breeding GAs. A natural breeding GA with the same structure is also developed and used as the reference for this evaluation. Then, an extensive numerical experiment on total of 2250 randomly generated scenarios is conducted to compare the effects of selective breeding mechanism. The effects of the varieties factors such as the number of jobs, the number of machines, the type of the initially selected parent and the different value of the crossover rates on the solution of the proposed algorithms are analyzed by the factorial ANOVA. A broad computational experiment is also conducted on the mean values of the solutions obtained by the proposed algorithms. In this experiment, the initial solution and the final solution of each algorithm is compared their associated percent improvements are calculated. The effect of the input parameters on the percent improvement as well as the improvement of different proposed algorithms are determined and compared. The computational results reveal that a significant improvement can be reached if one employs an initial population with better genes as one of the parents. It is also revealed that using an initial population with better genes as one of the parents performs much better when we let the initial selected parent is also being evolved throughout the progress of evolution.

Additionally, evaluation of the deviations of the solution of the proposed algorithms from the optimal solution disclose that all the proposed algorithms provide the near optimal solutions, while Algorithm 3 performs much better than the other two-proposed algorithms.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Vairaktarakis G, Emmons H. Flow shop scheduling: Theoretical results, algorithms, and applications. Springer, USA; 2013.

[2] Border P. Improving livestock. Post Note, Houses of Parliament, Parliamentary Office of Science and Technology. 2011;393.

[3] Ghassemi-Tari F, Olfat L. Two COVERT based algorithms for solving the generalized flow-shop problem. 34th International Conference on Computers and Industrial Engineering, San Francisco, California; 2004.

[4] Ghassemi-Tari F, Olfat L. Development of a set of algorithms for the multi-projects scheduling problems. Journal of Industrial and Systems Engineering. 2007;1:11-17.

[5] Ghassemi-Tari F, Olfat L. COVERT based algorithms for solving the generalized tardiness flow-shop problems. Journal of Industrial and Systems Engineering. 2008; 2: 197-213.

[6] Ghassemi-Tari F, Olfat L. A Set of algorithms for solving the generalized tardiness Flow shop problems. Journal of Industrial and Systems Engineering, 2010;4:156-166.

[7] Ghassemi-Tari F, Olfat L. Heuristic rules for tardiness problem in Flow shop with intermediate due dates. International Journal of Advanced Manufacturing Technology. 2013;71:381-393.

[8] Ghassemi Tari F, Alaei R. Scheduling TV commercials using genetic algorithms. International Journal of Production Research. 2013;51(6):4921-4929.

[9] Holland JA. Adaptation in natural and artificial systems. Dissertation, University of Michigan, Ann Arbor; 1975.

[10] Werner F. On the heuristic solution of the permutation flow shop problem by path algorithms. Computers & Operations Research. 1993;20:707–722.

[11] Murata T, Ishibuchi H, Tanaka H. (1996) GAs for flow shop scheduling problems. Computers & Industrial Engineering. 2013;30:1061-1071.

[12] Zheng D, Wang L. An effective hybrid heuristic for flow shop scheduling. International Journal of Advanced Manufacturing Technology. 2003;21(1):38-44.

[13] Wang L, Zhang L. Determining optimal combination of genetic operators for flow shop scheduling. International Journal of Advanced Manufacturing Technology. 2006;30(3):302–308.

[14] Chung C.S, Flynn J, Rom W, Stalinski P. A genetic algorithm to minimize the total tardiness for m-machine permutation flow shop problems. Journal of Entrepreneurship: Management and Innovation. 2012;8:26-43.

[15] Cemil IM, Bilal T, Veli C. Scheduling in a two-machine flow-shop for earliness/tardiness under learning effect. International Journal of Advanced Manufacturing Technology. 2012;61:1129-1137.

[16] Vallada E, Ruiz R. Genetic algorithms with path relinking for the minimum tardiness permutation flow shop problem. Omega. 2010;38:57-67.

[17] Onwubolu GC, Mutingi M. Genetic algorithm for minimizing tardiness in flow-shop scheduling. Production Planning and Control. 1999;10(5):462-471.

[18] Babu PM, Sai BVH, Reddy AS. Optimization of make-span and total tardiness for flow shop scheduling using genetic algorithm. International Journal of Engineering Research and General Science. 2015;3(3):195-199.

[19] Ta QC, Billaut JC, Bouquard JL. Metaheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. J Intelligent Manufacturing Published on line on 5 February 2015;1-12.

[20] Rahman HF, Sarker R, Essam D. A genetic algorithm for permutation flow shop scheduling under make to stock production system. Computers & Industrial Engineering. 2015;90:12-24.

[21] Sajadi SM, Kashan AH, Khaledan S. A new approach for permutation flow-shop scheduling using league championship algorithm. CIE44 & IMSS' 14-16, October 2014, Istanbul Turkey. 2014: 1-4.

[22] Shin S. A modified genetic algorithm for the flow-shop scheduling problem. Academy of Information and Management Sciences Journal. 2013;16:7-14.

[23] Wang F, Rao YQ, Wang F, Hou Y. Design and application of a new hybrid heuristic algorithm for flow shop scheduling. International Journal of Computer Network and Information Security. 2011;3: 41-49.

[24] Cui WW, Lu Z, Zhou B, Li C, Han X. A hybrid genetic algorithm for non-permutation flow shop scheduling problems with unavailability constraints. International Journal of Computer Integrated Manufacturing. 2016;29(9):944-961.

[25] Işler MC, Toklu B, Çelik V. Scheduling in a two-machine flow-shop for earliness/tardiness under learning effect. International Journal of Advanced Manufacturing Technology. 2012;61(9):1129–1137.

[26] Lee WC, Chung YH. Permutation flowshop scheduling to minimize the total tardiness with learning effects. International Journal of Production Economics. 2013;141(1):327-334.

[27] Kia H, Ghodsipour SH, Davoudpour H. New scheduling rules for a dynamic flexible flow line problem with sequence-dependent setup times. Journal of Industrial Engineering International, First Online: 19 January; 2017.

[28] Mou J, Gao L, Guo Q, Mu J. A hybrid heuristic algorithm for flowshop inverse scheduling problem under a dynamic environment. Cluster Computing . 2017;20(1):439–453.

[29] Meshkinfam S, Ghassemi Tari F. A genetic algorithm for generalized tardiness flowshop problems. International Journal of Engineering Science and Technology. 2016;8(9):219-228.

Appendix 1

The results of the factorial-ANOVA

Table A1-1. The result of 3-factorial (*n, m, CR*) ANOVA for Algorithm 1

Dependent variable: Result					
Source	**DF**	**Sum of squares**	**Mean squares**	**F value**	**Pr>F**
Model	149	1. 93176136	0. 01296484	10.10	<0.0001
Error	600	0. 77053880	0. 00128423		
Corrected	749	2. 70230016			
	R-Square	**Coefficient variance**	**Root MSE**	**Result mean**	
	0. 714858	39. 37394	0. 035836	0. 091015	
Source	**DF**	**ANOVA SS**	**Mean square**	**F value**	**Pr > F**
A	5	1. 60148556	0. 32029711	249.41	<.0001
B	4	0. 17423186	0. 04355797	33.92	<.0001
C	4	0. 00680253	0. 00170063	1.32	0.2595
A*B	20	0. 06209585	0. 00310479	2.42	0.0002
A*C	20	0. 01407101	0. 00070355	0.55	0.9457
B*C	16	0. 01228248	0. 00076766	0.60	0.8867
A*B*C	80	0. 06079206	0. 00075990	0.59	0.9979

Table A1-2. The result of 3-factorial (n, m, *CR*) ANOVA for ISP = STP of Algorithm 2

Dependent variable: Result						
Source		**DF**	**Sum of squares**	**Mean squares**	**F value**	**Pr>F**
Model		149	2. 13594515	0. 01433520	12.53	<0.0001
Error		600	0. 68626469	0. 00114377		
Corrected total		749	2. 82220984			
	R-square	**Coefficient variance**	**Root MSE**	**Result mean**		
	0. 756834	36. 80532	0. 033820	0. 091888		
Source	**DF**	**ANOVA SS**	**Mean square**	**F value**		**Pr > F**
A	5	2. 01130297	0. 40226059	351.70		<.0001
B	4	0. 04538443	0. 01134611	9.92		<.0001
C	4	0. 00155339	0. 00038835	0.34		0.8513
A*B	20	0. 04099177	0. 00204959	1.79		0.0184
A*C	20	0. 00767002	0. 00038350	0.34		0.9974
B*C	16	0. 00363802	0. 00022738	0.20		0.9997
A*B*C	80	0. 02540455	0. 00031756	0.28		1.0000

Table A1-3. The result of 3-factorial (*n, m, CR*) ANOVA for *ISP* = EDD of Algorithm 2

Dependent variable: Result						
Source		**DF**	**Sum of squares**	**Mean squares**	**F value**	**Pr>F**
Model		149	2. 02048040	0. 01356027	18.08	<0.0001
Error		600	0. 44990766	0. 00074985		
Corrected total		749	2. 47038806			
	R-square	**Coefficient variance**	**Root MSE**	**Result mean**		
	0. 817880	29. 84111	0. 027383	0. 091764		
Source	**DF**	**ANOVA SS**	**Mean square**	**F value**		**Pr > F**
A	5	1. 82850990	0. 36570198	487.70		<.0001
B	4	0. 11460198	0. 02865049	38.21		<.0001
C	4	0. 00101875	0. 00025469	0.34		0.8512
A*B	20	0. 05779026	0. 00288951	3.85		<.0001
A*C	20	0. 00238333	0. 00011917	0.16		1.0000
B*C	16	0. 00313348	0. 00019584	0.26		0.9985
A*B*C	80	0. 01304270	0. 00016303	0.22		1.0000

Table A1-4. The result of 3-factorial (n, m, *CR*) ANOVA for ISP = STP-EDD of Algorithm 2

Dependent variable: Result						
Source		**DF**	**Sum of squares**	**Mean squares**	**F value**	**Pr>F**
Model		149	2. 33695266	0. 01568425	13.13	<0.0001
Error		600	0. 71659846	0. 00119433		
Corrected total		749	3. 05355112			
	R-Square	**Coefficient variance**	**Root MSE**	**Result mean**		
	0. 765323	35. 44298	0. 034559	0. 097506		
Source	**DF**	**ANOVA SS**	**Mean square**	**F value**		**Pr > F**
A	5	2. 12443108	0. 42488622	355.75		<.0001
B	4	0. 09668035	0. 02417009	20.24		<.0001
C	4	0. 00322906	0. 00080726	0.68		0.6089
A*B	20	0. 06134522	0. 00306726	2.57		0.0002
A*C	20	0. 00736230	0. 00036811	0.31		0.9986
B*C	16	0. 00455956	0. 00028497	0.24		0.9991
A*B*C	80	0. 03934509	0. 00049181	0.41		1.0000

Table A1-5. The result of 3-factorial (n, m, *CR*) ANOVA for ISP = STP of Algorithm 3

Dependent variable: Result						
Source		**DF**	**Sum of square**	**Mean squares**	**F Value**	**Pr>F**
Model		149	0. 74745810	0. 00501650	9.07	<0.0001
Error		600	0. 33176744	0. 00055295		
Corrected Total		749	1. 07922554			
	R-square	**Coefficient variance**	**Root MSE**	**Result mean**		
	0. 692587	16. 44972	0. 023515	0. 142949		
Source	**DF**	**ANOVA SS**	**Mean square**	**F value**		**Pr > F**
A	5	0. 65865082	0. 13173016	238.23		<.0001
B	4	0. 03855183	0. 00963796	17.43		<.0001
C	4	0. 00003834	0. 00000958	0.02		0.994
A*B	20	0. 04221587	0. 00211079	3.82		<.0001
A*C	20	0. 00180286	0. 00009014	0.16		1.0000
B*C	16	0. 00055526	0. 00003470	0.06		1.0000
A*B*C	80	0. 00564311	0. 00007054	0.13		1.0000

Table A1-6. The result of 3-factorial (n, m, CR) ANOVA for ISP = EDD of Algorithm 3

Dependent variable: Result					
Source	**DF**	**Sum of squares**	**Mean squares**	**F value**	**Pr>F**
Model	149	2. 02637112	0. 01359981	16.31	<0.0001
Error	600	0. 50043850	0. 00083406		
Corrected Total	749	2. 52680962			

R-square	**Coefficient variance**	**Root MSE**	**Result mean**
0. 801948	31. 38935	0. 028880	0. 092006

Source	**DF**	**ANOVA SS**	**Mean square**	**F value**	**Pr > F**
A	5	1. 53819704	0. 30763941	368.84	<.0001
B	4	0. 34942983	0. 08735746	31048.74	<.0001
C	4	0. 00204869	0. 00051217	0.61	0.6527
A*B	20	0. 09972464	0. 00498623	5.98	<.0001
A*C	20	0. 00672956	0. 00033648	0.40	0.9909
B*C	16	0. 00514515	0. 00032157	0.39	0.9857
A*B*C	80	0. 02509621	0. 00031370	0.38	1.0000

Table A1-7. The result of 3-factorial (n, m, *CR*) ANOVA for ISP = STP-EDD of Algorithm 3

Dependent variable: Result					
Source	**DF**	**Sum of squares**	**Mean squares**	**F value**	**Pr>F**
Model	149	0. 95166631	0. 00638702	9.15	<0.0001
Error	600	0. 41879107	0. 00069799		
Corrected Total	749	1. 37045738			

R-square	**Coefficient variance**	**Root MSE**	**Result mean**
0. 694415	35. 53341	0. 026419	0. 074351

Source	**DF**	**ANOVA SS**	**Mean square**	**F value**	**Pr > F**
A	5	0. 69707164	0. 13941433	199.74	<.0001
B	4	0. 18395334	0. 04598833	65.89	<.0001
C	4	0. 00204856	0. 00051214	0.73	0.5692
A*B	20	0. 04093752	0. 00204688	2.93	<.0001
A*C	20	0. 00506512	0. 00025326	0.36	0.9955
B*C	16	0. 00508895	0. 00031806	0.46	0.9663
A*B*C	80	0. 01750118	0. 00021876	0.31	1.0000

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/19284