International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

An effective genetic algorithm with a critical-path-guided Giffler and Thompson crossover operator for job shop scheduling problem

Mohamed Kurdi*[1]

Submitted: 14/05/2018 Accepted : 25/02/2019 Published: 20/03/2019

Abstract: This work presents an effective genetic algorithm with a critical-path-guided Giffler and Thompson crossover operator for job shop scheduling problem with the objective of makespan minimization (GA-CPG-GT). Even though passing important traits from parents to offspring is known to be an important feature of any uniform crossover operator, most of the proposed operators adopt random exchange of genetic materials between parents; this is probably due to the fact that it is tricky to identify the genetic materials that hold the important traits. For that reason, in this work, a new selective exchange of genetic materials is proposed. In the proposed approach, at first, the genetic materials that hold the important traits are identified according to some domain specific information provided by the critical paths of the parents, and then the exchange is made on favor of them. The properties of critical path are usually utilized by the local search methods such as tabu search and simulating annealing; however, in this work, they are utilized in the global search method GA during the crossover operator. The implications of the proposed approach are investigated using the Giffler and Thompson crossover operator, which is a uniform crossover combined with the G&T algorithm. The proposed approach is tested on 55 benchmark instances, with the proposed selective exchange, and without it using the random one, and also compared with other 5 similar works reported in the literature. The computational results validate the enhancements accomplished by the proposed selective exchange, and show the superiority of the proposed algorithm over the compared works in terms of solution quality, and validate its effectiveness.

Keywords: *Critical path, Genetic algorithm, Job shop scheduling, Metaheuristic, Uniform crossover.*

1. Introduction

Job shop scheduling problem (JSSP) is an NP-hard problem, and one of the most difficult combinatorial optimization problems considered to date. Due to its difficulty of solving and importance for production management with regard to enhancing machine utilization and shortening cycle-time, JSSP has been tackled by so many methods over more than fifty years. The methods used range from exact methods [1], to heuristic [2] [3], and finally to metaheuristics such as simulated annealing (SA) [4], tabu search (TS) [5], ant colony optimization (ACO) [6], parallel artificial bee colony optimization (ABCO) [7], discrete particle swarm optimization (PSO) [8], modified clonal selection algorithm (CSA) [9], and parallel bat algorithm (BA) [10]. An overview of JSSP techniques can be found in Zobolas et al. [11], while an outdated but comprehensive survey of them can be found in Jain and Meeran [12].

Genetic algorithm (GA) is a well-known global search method that has a wide range of applications for solving combinatorial optimization problems. Regarding JSSP, many GAs with various approaches have been developed, the most common ones include developing encoding and decoding schemes [13], developing genetic operators [14], hybridizing with other algorithms [15], and designing parallel GA (PGA) models [16] [17] .

The crossover operator is one of the main components of GA because it provides the exploitation of search space by enabling pairs of solutions (parents) to mate and generate new solutions (offspring) by mutual exchange and recombination of their genetic materials; therefor, several crossover operators have been developed for JSSP, such as partial-mapped crossover (PMX), order crossover (OX), cycle crossover (CX), position-based crossover, order-based crossover, etc. An overview of them can be found in [18]. Unlike the other types of crossover operators, the Giffler and Thompson (GT) based crossover operators are problem dependent operators, which are distinguished in the ability to interact in the phenotype space of individuals, and thus produce active schedules directly without the requirements of decoding and repairing procedures. This ability is provided by utilizing the principles of the conflict sets defined by G&T algorithm, which is an algorithm for generating active schedules (a subclass of the search space that includes the optimal solutions), and usually combined with dispatching rules for the creation of an initial population. The first GT based crossover operator is called GA-GT crossover, and was proposed by Yamada and Nakano [19]. In their approach, a direct representation in the form of a string of the operation completion times of an active schedule is used, and the GA-GT crossover can be described as follows. At each decision point, one parent is chosen randomly, but the operation that is schedulable and has the earliest completion time in the parent schedule is scheduled next. This strategy corresponds to a uniform crossover combined with the G&T algorithm as an interpreter to convert illegal offspring into feasible active schedules. Peng and Salim [20] proposed a modified GA-GT crossover, in which the major modification is using a binary tournament selection for selecting the parents (instead of random pairing applied in original GA-GT crossover). Moonen and Janssens [21] proposed a new crossover called Giffler-Thompson Focused (GTF) crossover which combines an order-based crossover with a one-point crossover, and uses the largest conflict set to direct the option of the cut point [22].

[1] *Aydın Adnan Menderes University, Computer Engineering Department, 09010, Aydin, TURKEY*
ORCID: 0000-0002-1461-1174
** Corresponding Author Email: mohamed.kurdi@adu.edu.tr*

Even though an important feature of any crossover operator is to be able to pass important genetic materials (traits) to offspring, most of the proposed uniform crossover operators that can be found in the literature, including all the aforementioned GT based crossovers, adopt random exchange of genetic materials between parents, which may decrease the exploitation of the search space because the resulting offspring may not inherit the important traits from their parents. This is maybe due to the fact that it is difficult to identify the genetic materials that hold the important traits of an individual. However, a recent study, made by Kurdi [23], has recommended identifying the important genetic materials according to some criteria and giving them the preference for exchange over the rest. In his work, he proposed an informed uniform crossover that employs the history of parents' evolution occurred during the self-adaptation phase (local search via TS) in determining the genetic materials exchanged between them i.e. giving preference to the genes that have been evolved recently. The findings of the aforesaid study, and the fact that only the genes belonging to the critical path can evolve (during the self-adaptation phase) have inspired us to propose a new selective exchange of genetic materials, which may handle the aforementioned shortcoming of the random one, and thus improve the exploitation of the search space. In the proposed approach, at first, the genetic materials that hold the important traits are identified according to some domain specific information provided by the critical paths of the parents, and then the exchange is made on favor of them. To the best of our knowledge, the concept of critical path is usually utilized only in local search methods (such as TS and SA), and this is the first work that utilizes it in the global search method GA during the crossover phase for selecting the genetic materials exchanged between individuals. Because of its efficiency and suitability, the GA-GT crossover operator [19] is used to study the influence of the proposed criteria.

2. Problem Definition

The classical JSSP with the objective of minimization of makespan consists of scheduling a set of n jobs $\{J_j\}\, 1 \le j \le n$ on a set of m machines $\{M_r\}\, 1 \le r \le m$. The processing of job J_j on machine M_r is called the operation O_{jr}, and lasts for a continued specific period called processing time P_{jr} (pre-emption is not permitted). Two constraints exist in the problem: the precedence constraint that states that each job J_j must be processed on each machine M_r according to a predetermined sequence called topological sequence, and the capacity constraint that enforces that each machine M_r must process only one job J_j at a time. The start time and completion time of operation O_{jr} are denoted as S_{jr}, C_{jr} respectively. A solution (or schedule) is defined as the set of the completion times for all operations; a feasible schedule is a schedule that fulfils the two constraints. The time required for the completion of all the jobs is called makespan and denoted as C_{max}, where C_{max} = max $_{1\le j\le n, 1\le r\le m}$ C_{jr}. The objective of the problem becomes finding a feasible solution that provides the minimum value of C_{max} [19]. An example of a 2 x 3 JSSP is given in Table 1. The data include the topological sequence of all jobs with their processing times. For example, job 1 is processed in this sequence $O_{11} \rightarrow O_{13} \rightarrow O_{12}$, i.e. it is processed on machine 1 for 3 time units, then on machine 3 for 3 units, then on machine 2 for 4 units. A possible solution of the 2 x 3 JSSP represented by a Gantt chart is given in Fig. 1.

Table 1. An example of a 2 × 3 JSSP.

Job	Machine / Processing time		
J1	M1 / 3	M3/ 3	M2/ 4
J2	M1 / 4	M2/ 6	M3/ 3

As shown in Fig. 2, the 2 x 3 JSSP problem can be also represented by the job sequence matrix $\{T_{jk}\}$ and processing time matrix $\{P_{jk}\}$; and its solution can be represented by a solution matrix $\{S_{rk}\}$. Where $T_{jk} = r$ means that k-th operation for job J_j is processed on machine M_r for P_{jk} time units, and $S_{rk} = j$ means that the k-th operation on machine M_r is job J_j.

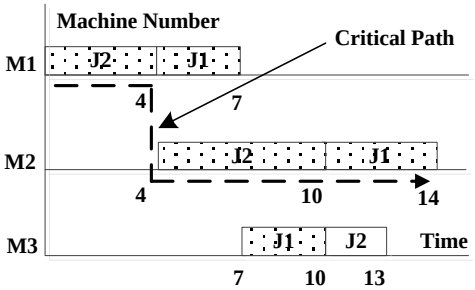

Fig. 1. A Gantt chart representation of a solution for the 2 × 3 problem.

$$\{T_{jk}\} = \begin{vmatrix} 1 & 3 & 2 \\ 1 & 2 & 3 \end{vmatrix},\ \{P_{jk}\} = \begin{vmatrix} 3 & 3 & 4 \\ 4 & 6 & 3 \end{vmatrix},\ \{S_{rk}\} = \begin{vmatrix} 2 & 1 \\ 2 & 1 \\ 1 & 2 \end{vmatrix}$$

Fig. 2. Matrix representation for the 2 × 3 problem and its solution.

The critical path is a sequence of critical operations (i.e. operations that has zero slack time) that has the longest length in the schedule [24]. In Fig. 1, an example of a possible critical path is indicated by the dashed line, whereas the critical operations are indicted by the dotted background.

3. The Proposed GA-CPG-GT Algorithm

At first, a set of individuals called population is created using the G&T algorithm. Each individual has two representations called phenotype and genotype. Whereas the phenotype represents behavioral traits of this individual in its environment (what an individual looks like) i.e. a potential solution to JSSP, the genotype represents the genetic composition of this individual in the form of a chromosome. The next step involves evaluating the fitness of each individual that measures its suitability for the surrounding environment i.e. how good the solution represented by it is for the JSSP. And then, the natural evolution of the population takes place through a series of generations [25]. At each generation, phases of cooperation evolutions alternate with phases of self-adaptation ones. While cooperation phases provide exploitation of the search space and mean that individuals evolve by exchanging their knowledge about the search space (inheriting acquired traits) via the crossover operator, self-adaptation phases provide exploration of the search space and mean that they evolve independently by using only their own knowledge (generating new traits) via the mutation operator [26]. The general methodology of the proposed algorithm is described in Fig. 3.

3.1. Initial Population

The initial population can be created by many ways such as G&T algorithm, priority dispatching rules, and random methods. Generally, the initial population creation methods have small effects on solution quality, but they may affect the running time

[27]. However, in this work, the G&T algorithm was used to create the initial population.

3.2. Chromosome Representation

The preference-list representation is used [28]. In this representation, for a problem of the size *n* x *m*, a chromosome is formed of *m* subchromosomes, each for one machine. Each subchromosome is a string of symbols with a length of *n*, and each symbol represents an operation processed on the relevant machine. For example, the solution of the 2 x 3 problem given in Fig. 1 is encoded in this form [(2 1) (2 1) (1 2)]. These subchromosomes are usually generated randomly and may conflict with the problem constraints (represent infeasible individuals); because of that, initially they are considered as preference lists and passed to a decoding (and repairing) procedure, that selects the operations that appear first for processing on the related machines, and also alters the sequence of these operations when it is necessary for meeting the problem constraints [13]. However, in this work, there is no need for the application of the decoding procedure on the chromosomes that are constructed during the creation of the initial population or that result from crossover, this is because these chromosomes and their solutions (genotypes and their corresponding phenotype) are constructed together using the G&T algorithm. Therefore, the decoding procedure is only required for the mutated individuals.

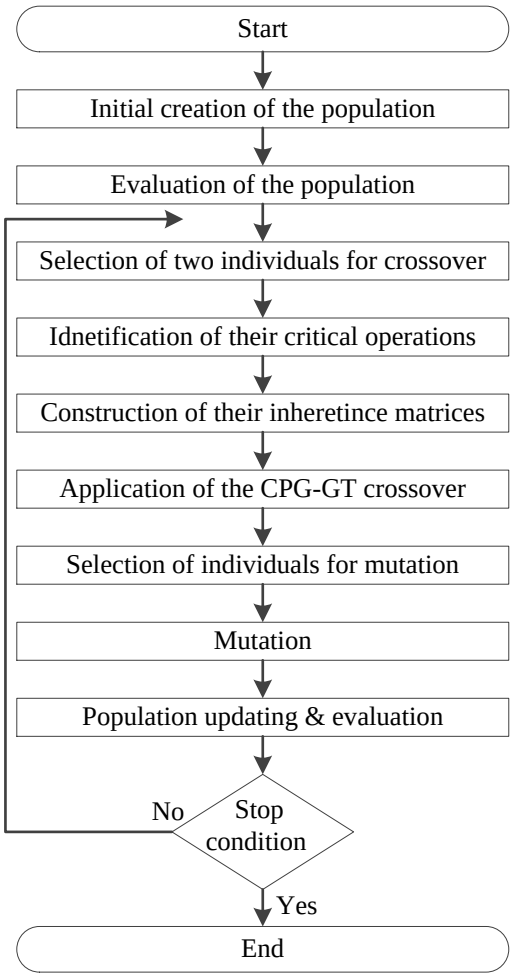

Fig. 3. The general methodology of the GA-CPG-GT algorithm.

3.3. Fitness Function

To calculate the survival probability of an individual at the upcoming generation, a fitness function is utilized to determine how good the solution represented by an individual is. In this work, the well-known function suggested by Goldberg [29] is used. It is defined by the following formula $F(x)=C_{maxP} - C_{max}(x)$, where C_{maxP} is the maximum makespan value found in the population under consideration, and $C_{max}(x)$ is the makespan value of the individual x.

3.4. Reproduction

Offspring are created through the application of crossover and/or mutation operators. The candidates for the crossover operator are selected using the roulette wheel method Goldberg [29], in which individuals are selected randomly with their probability of selection proportional to their relative fitness in the population. Thus, fitter individuals have a greater chance to survive and reproduce than the weaker ones. If the population size is *n*, the probability of selection of an individual x_j can be calculated by the following formula.

$$P(x_j) = F(x_j) / \sum_{i=1}^{n} F(x_i) \quad (1)$$

The candidates for mutation crossover are the worst ones with a hope of introducing better traits to them, thereby increasing their chances of survival.

3.5. The Proposed CPG-GT Crossover Operator

The proposed CPG-GT crossover operator is described in Algorithm 1. The main difference between it and the classical GA-GT proposed by Yamada and Nakano [19] is in the (additional) Step 2 i.e. in the generation of the inheritance matrices. In their work, for each pair of parents p_0 and p_1 a binary matrix *H* of the size *n* x *m* called inheritance matrix is generated randomly with equal probability of 0 and 1; the purpose of this matrix is to determine the genetic material exchanged between parents as follows: $H_{ri} = 0$ means that the *i*-th operation on machine *r* should be determined by the first parent p_0, and $H_{ri} = 1$ means that it will be determined by the second parent p_1; and by this way, the first child is produced, while the second child is produced by switching the roles of p_0 and p_1 (for an example of the classical GA-GT please refer to [24]). However, due to the randomness in the creation of *H*, the resulting offspring are expected to inherent random traits of their parents; consequently, this may weaken the exploitation of search space provided by the crossover operator. In order to overcome this drawback, in this work, the inheritance matrix is initialized in the light of the critical path. The basic assumption is that the critical operations, which are the building units of any possible critical path, possess

Algorithm 1 (CPG-GT crossover operator)

A scheduling problem represented by the job sequence matrix $\{T_{jk}\}$, and the processing time matrix $\{P_{jk}\}$ as well as two solution schedules p_0 and p_1 represented by solution matrices $S^0 = \{S^0_{rk}\}$ and $S^1 = \{S^1_{rk}\}$ respectively, are given as inputs.

1. Initialize *G* as a set of operations that are first in the job sequence; i.e., $G = \{O_{1T11}, O_{1T21}, \ldots, O_{1Tn1}\}$. For each operation $O \in G$, set *the earliest starting time* $ES(O) = 0$ and *the earliest completion time* $EC(O) = p(O)$.
2. Generate the binary inheritance matrices *H* for p_0, using Algorithm 2.
3. Find the earliest completable operation (whose earliest completion time is the smallest) $O_{*r} \in G$ as follows. O_{*r} = arg min $\{EC(O)| \in G\}$ with machine M_r. A subset of *G* that consists of operations processed on machine M_r is denoted as *Gr*.

4. Calculate the conflict set $C[M_r; i] \subset Gr$ as follows. $C[M_r; i] = \{O_{kr} \in G \mid ES(O_{kr}) < EC(O_{*r})\}$, where i-1 is the number of operations that are already scheduled on M_r.

5. Select one of the parents $\{p_0, p_1\}$ as p according to the value of H_{ri}, that is, $p = p_{Hri}$ and $S^p = S^{Hri}$. For each $Ojr \in C[M_r; i]$ with job number j, there exists an index l such that $S_{rl} = j$. Let l_m be the smallest index number among them; i.e., $l_m = \min \{l \mid S_{rl} = j \text{ and } O_{jr} \in C[Mr; i]\}$ and let $k = S_{rlm}$. This results in selecting an operation $O_{kr} \in C[M_r; i]$ that has been scheduled in p earliest among the members of $C[M_r; i]$.

6. Schedule O_{kr} as the i-th operation on M_r; i.e. $S_{ri} = k$, with its starting and completion times equal to $ES(O_{kr})$ and $EC(O_{kr})$ respectively: $s(O_{kr}) = ES(O_{kr})$; $c(O_{kr}) = E(CO_{kr})$.

7. For all $O_{jr} \in G_r \backslash \{O_{kr}\}$, update $ES(O_{jr})$ as $ES(O_{jr}) = \max \{ES(O_{jr}); EC(O_{kr})\}$ and $EC(O_{jr})$ as $EC(O_{kr}) = ES(O_{kr}) + p(O_{kr})$.

8. Remove O_{kr} from G (and therefore from G_r), and add operation O_{ks} that is the next to O_{kr} in the job sequence to G if such O_{ks} exits; i.e., if $r = T_{ki}$ and $i < m$, then $s = T_{ki+1}$ and $G = (G\backslash\{O_{kr}\}) \cup \{O_{ks}\}$. Calculate $ES(O_{ks})$ and $EC(O_{ks})$ as: $ES(O_{ks}) = \max \{EC(O_{kr}); EC(PM(O_{ks}))\}$ and $EC(O_{ks}) = ES(O_{ks}) + p(O_{ks})$ respectively.

9. Repeat from Step 3 to Step 8 until all operations are scheduled.

10. Output the solution matrix $\{S_{rk}\}$ as the active schedule obtained with the set of starting and completion times $\{s(O_{jr})\}$ and $\{c(O_{jr})\}$ respectively, where $j = S_{rk}$.

more important traits than the others and should be inherited by the offspring. Based on this assumption, 50% of them will be transferred to offspring, while the rest are taken randomly from the second parent. The generation of the inheritance matrix that implements this idea is given in Algorithm 2. It can be noted from Algorithm 2, that unlike the traditional GA-GT crossover operator, each parent will have its own inheritance matrix. Another difference between the proposed operator and the GA-GT is that the proposed one does not integrate the mutation operator in Step 5, and there is an independent procedure for it.

Algorithm 2 (Inheritance matrix generation)

```
Let S=[(s11,s12,…s1n),…( s21,s22,…s2n),…( sm1,sm2,…smn)] be the first parent.
Let ST(Ojr) be the slack time of the operation Ojr.
for x=1,n do
  for y=1,m do
    j=Sxy; r=x;
    if (ST(Okr)==0)
      r = a random value in the range [0, 1];
      if (r <= 50) Hij=1;
      else Hij=0;
    else H0ij=0;
  end for
end for
```

3.6. Mutation

Mutation is the process of randomly changing the values of genes in a chromosome. The main objective of it is to introduce new genetic materials into some individuals in the population, thereby promote the exploration of search space and avoid the premature convergence [25]. The classical mutation operators include inversion, insertion, and swap [30]. In this study, the inversion operator is adopted. This operator acts as follows. It selects two genes randomly and then inverts the substring that exists between these two genes. Since the mutated chromosomes may represent infeasible individuals, their associated solutions are generated by the decoding procedure discussed in Section 3.2.

3.7. Replacement Strategy and Stop Condition

A steady-state generation replacement method with elitist strategy [29] is adopted. An old generation is not entirely replaced by the new one; the best individual is copied to the next generation without any changes. The stop condition is either the best known solution has been reached, or the maximum number of generations has been elapsed.

4. Computational Results

The algorithm was implemented in C++, and the tests were run on a PC with 3.40 GHz Intel(R) Core (TM) i7-3770 CPU and 8.00 GB. The parameters were tuned through a number of experiments; as a result, they were fixed as follows: the population size 100, the maximum number of generations 1000, t crossover probability 0.8, and mutation probability 0.05.

In order to validate the effectiveness of the proposed algorithm, it was tested on 55 instances from four classes of standard JSSP benchmark instances: Fisher and Thompson [31] instances ft06, ft10, ft20; Lawrence [32] instances la01–la40; Applegate and Cook [33] instances orb01–orb09; and two of Adams et al. [2] instances denoted as ABZ5 and ABZ6.

GA-CPG-GT was compared with another version of it that is identical to it, but uses a random initialization of the inheritance matrix with equal probability of 0 and 1 (as proposed by Yamada and Nakano [19]), and also compared with other 5 similar works found in the literature: an agent-based parallel genetic algorithm PaGA [34], ant colony optimization with parameterized search space (ACO-PA) [35], multiple independent particle swarms algorithm JSP/PSO [36], hybrid parallel micro genetic algorithm (HPGA) models [16], and modified clonal selection algorithm (CSA) [9]. The best makespan obtained by the proposed algorithm from 10 independent runs was used as a criterion of the performance.

Table 2 summarizes the experimental results on the 55 instances, it lists problem name, problem size (number of jobs × number of machines), the best known solution (BKS); and the best solution (BS) obtained by each of the compared algorithms.

From Table 2, it can be seen that GA-CPG-GT is able of reaching equal or better results than the compared works on almost the whole set of instances; however, to make a precise comparison, the relative deviation of the best solution was calculated using the following formula.

$$\text{BS-RD} = 100 \times (\text{BS-BKS})/\text{BKS} \quad (2)$$

The average value of BS-RD, denoted as BS-ARD, was also calculated for each algorithm over its instances. Table 3 shows the number of instances solved (NIS), BS-ARD values for GA-CPG-GT and the other algorithms (OA), the relative improvement achieved by GA-CPG-GT in BS-ARD values with respect to each of the other algorithms. From Table 3 and Fig. 4, it can be noticed that GA-CPG-GT yields remarkably significant relative improvement to all of the other compared algorithms.

In comparison with the classical GA-R-GT that identical to GA-CPG-GT but adopts random exchange of genetic materials, the relative improvement is 53%, which validates the assumptions made about the proposed selective exchange of genetic materials

in promoting the exploitation of the search space. From the other hand, it can be seen that GA-CPG-GT could outperform all of the other compared works in terms of solution quality; the improvement made by it reaches up to 96% when it is compared with ACO-PA, which means that the proposed algorithm can be considered as an effective approach for solving JSSP.

Table 2. Comparison of GA-CPG-GT with the other algorithms.

Problem			GA-CPG-GT	GA-R-GT	PaGA	HPGA	ACO-PA	MPSO	MCSA
Name	**Size**	**BKS**	BS	BS	BS	BS	BS	BS	BS
ft06	6x6	55	**55**	**55**	**55**	-	-	57	**55**
ft10	10x10	930	935	969	997	931	-	956	1034
ft20	20x5	1165	1180	1199	1196	**1165**	-	1180	-
orb01	10x10	1059	1084	1087	1149	1085	-	-	-
orb02	10x10	888	890	906	929	-	-	-	-
orb03	10x10	1005	1037	1109	1129	-	-	-	-
orb04	10x10	1005	1028	1066	1062	1054	-	-	-
orb05	10x10	887	894	903	936	-	-	-	-
orb06	10x10	1010	1035	1063	1060	-	-	-	-
orb07	10x10	397	404	407	416	-	-	-	-
orb08	10x10	899	937	926	1010	-	-	-	-
orb09	10x10	934	943	944	994	-	-	-	-
orb10	10x10	944	967	987	-	-	-	-	-
abz5	10x10	1234	1238	1280	-	-	-	-	1270
abz6	10x10	943	947	975	-	-	-	-	943
la01	10x5	666	**666**	**666**	**666**	-	**666**	**666**	**666**
la02	10x5	655	**655**	667	**655**	680	665	668	**655**
la03	10x5	597	**597**	626	617	-	609	606	**597**
la04	10x5	590	**590**	595	607	-	597	611	**590**
la05	10x5	593	**593**	**593**	**593**	-	**593**	**593**	**593**
la06	15x5	926	**926**	**926**	**926**	-	**926**	**926**	-
la07	15x5	890	**890**	**890**	**890**	-	**890**	**890**	-
la08	15x5	863	**863**	**863**	**863**	-	**863**	**863**	-
la09	15x5	951	**951**	**951**	**951**	-	**951**	**951**	-
la10	15x5	958	**958**	**958**	**958**	-	**958**	**958**	-
la11	20x5	1222	**1222**	**1222**	**1222**	-	**1222**	**1222**	-
la12	20x5	1039	**1039**	**1039**	**1039**	-	**1039**	**1039**	-
la13	20x5	1150	**1150**	**1150**	**1150**	-	**1150**	**1150**	-
la14	20x5	1292	**1292**	**1292**	**1292**	-	**1292**	**1292**	-
la15	20x5	1207	**1207**	**1207**	**1207**	-	1251	**1207**	-
la16	10x10	945	946	979	994	947	995	988	1024
la17	10x10	784	**784**	804	793	-	786	792	-
la18	10x10	848	**848**	865	860	-	**848**	860	-
la19	10x10	842	**842**	876	873	-	856	875	-
la20	10x10	902	907	911	912	-	930	938	-
la21	15x10	1046	1090	1136	1146	1067	-	1082	-
la22	15x10	927	954	1003	1007	-	-	977	-
la23	15x10	1032	**1032**	1044	1033	-	-	**1032**	-
la24	15x10	935	974	983	1012	-	-	975	-
la25	15x10	977	999	1029	1067	-	-	1013	-
la26	20x10	1218	1237	1303	1323	-	-	1237	-
la27	20x10	1235	1313	1314	1359	-	-	1290	-
la28	20x10	1216	1280	1291	1369	-	-	1251	-
la29	20x10	1152	1247	1301	1322	1154	-	1247	-
la30	20x10	1355	1367	1393	1437	-	-	**1355**	-
la31	30x10	1784	**1784**	**1784**	1844	1906	-	**1784**	-
la32	30x10	1850	**1850**	**1850**	1907	-	-	**1850**	-
la33	30x10	1719	**1719**	1722	-	-	-	**1719**	-
la34	30x10	1721	1725	1766	-	-	-	1748	-
la35	30x10	1888	**1888**	**1888**	-	-	-	**1888**	-
la36	15x15	1268	1308	1361	-	1308	-	1332	-
la37	15x15	1397	1489	1485	-	-	-	1468	-
la38	15x15	1196	1275	1294	-	-	-	1280	-
la39	15x15	1233	1290	1327	-	-	-	1267	-
la40	15x15	1222	1252	1304	-	-	-	1286	-

Table 3. Relative improvements from GA-CPG-GT to the other works.

Algorithm	NIS	BS-ARD		RI_{BS-ARD}
		OA (%)	GA-CPG-GT (%)	GA-CPG-GT (%)
GA-R-GT	55	3,22	1,52	53
PaGA	44	4,27	1,29	70
HPGA	10	2,36	2,22	6
ACO-PA	20	0,93	0,03	96
MPSO	43	2,10	1,44	31
MCSA	10	2,25	0,14	94

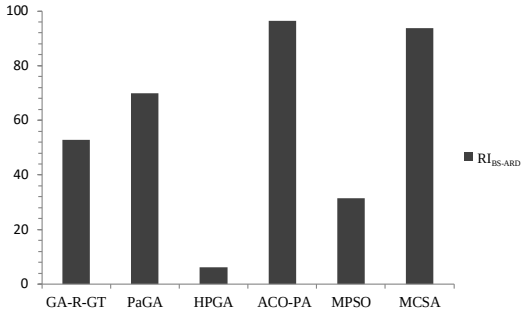

Fig. 4. Visualization of the relative improvements.

5. Conclusion and Further Works

Given the fact that an important feature of any uniform crossover is to enable the inheritance of important genetic materials (traits) from parent to their offspring, this study aims to study the impacts of selecting the genetic materials exchanged during crossover in the light of the domain specific information that exists in the critical path instead of selecting them randomly. The proposed algorithm has been tested on 55 benchmark instances, with the proposed selective exchange, and without it using the random exchange, and also compared with other 5 similar algorithms found in the literature. The experimental results validate the assumptions made about the proposed selective exchange of genetic materials in promoting the exploitation of the search space, and show the superiority of the proposed algorithm over the other compared works in terms of solution quality.

The basic idea proposed for the identification of the genes that hold the most important traits is a promising area of research, and it is worthy of further investigation on other combinatorial optimization problems since it yields significant improvements when applied on JSSP, which is one of the most difficult combinatorial optimization problems. Furthermore, the proposed approach is effective, simple to understand, and easy to implement; and these characteristics are sometimes welcome in real applications, especially industrial applications.

References

[1] J. Carlier, E. Pinson, "An algorithm for solving the job-shop problem", Manag Sci, vol. 35, no. 2, pp. 164-176, 1989. https://doi.org/10.1287/mnsc.35.2.164

[2] J. Adams, E. Balas, D. Zawack, "The shifting bottleneck procedure for job shop scheduling", Manag Sci, vol. 34, no. 3, pp. 391-401, 1988. https://doi.org/10.1287/mnsc.34.3.391

[3] J.H. Blackstone, D.T. Phillips, G.L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations", Int J Prod Res, vol. 20, no. 1, pp. 27-45, 1982. https://doi.org/10.1080/00207548208947745

[4] T. Satake, K. Morikawa, K. Takahashi, N. Nakamura, " Simulated annealing approach for minimizing the makespan of the general job-shop",Int J Prod Econ, pp. 60–61, pp. 515-522, 1999. https://doi.org/10.1016/S0925-5273(98)00171-6

[5] E. Nowicki, C. Smutnicki, "An advanced tabu search algorithm for the job shop problem", J Sched, vol. 8, no. 2, pp. 145-159, 2005. https://doi.org/10.1007/s10951-005-6364-5

[6] N. Fığlalı, C. Özkale, O. Engin, A. Fığlalı, "Investigation of ant system parameter interactions by using design of experiments for job-shop scheduling problems", Comput Ind Eng, vol. 56, no. 2, pp. 538-559, 2009. https://doi.org/10.1016/j.cie.2007.06.001

[7] L. Asadzadeh, "A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy", Comput Ind Eng, vol 102, pp. 359-367, 2016. https://doi.org/10.1016/j.cie.2016.06.025

[8] K. &. R. C. Rameshkumar, “A novel discrete PSO algorithm for solving job shop scheduling problem to minimize makespan”. In IOP Conference Series: Materials Science and Engineering, vol. 310, no. IOP Publishing, 2018. https://doi.org/10.1088/1757-899X/310/1/012143

[9] Atay, Y., & Kodaz, H. “Optimization of job shop scheduling problems using modified clonal selection algorithm”. Turk J Elec Eng & Comp Sci, vol. 22, no. 6, pp. 1528-1539, 2014. https://doi.org/10.3906/elk-1212-26

[10] Dao, T. K., Pan, T. S., & Pan, J. S.“Parallel bat algorithm for optimizing makespan in job shop scheduling problems”, J Intell Manuf, vol. 29, no. 2, pp. 451-462, 2018. https://doi.org/10.1007/s10845-015-1121-x

[11] G. Zobolas, C. Tarantilis, G. Ioannou, “Exact, heuristic and meta-heuristic algorithms for solving job shop scheduling problems”, F. Xhafa, A. Abraham (Eds.), Metaheuristics for scheduling in industrial and manufacturing applications, Springer, Berlin (2008), pp. 1-40. https://doi.org/10.1007/978-3-540-78985-7_1

[12] A.S. Jain, S. Meeran, “Deterministic job-shop scheduling: past, present and future”, Eur J Oper Res, vol. 113, no. 2, pp. 390-434, 1999. https://doi.org/10.1016/S0377-2217(98)00113-1

[13] R. Cheng, M. Gen, Y. Tsujimura A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation Comput Ind Eng, vol. 30, no. 4, pp. 983-997, 1996. https://doi.org/10.1016/0360-8352(96)00047-2

[14] M. Watanabe, K. Ida, M. Gen, "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem", Comput Ind Eng, vol. 48, no. 4, pp. 743-752, 2005 https://doi.org/10.1016/j.cie.2004.12.008

[15] M. Kurdi, “A new hybrid island model genetic algorithm for job shop scheduling problem”, Comput Ind Eng, vol. 88, 273-28, 2015. https://doi.org/10.1016/j.cie.2015.07.015

[16] R. Yusof, M. Khalid, G.T. Hui, S. Md Yusof, M.F. Othman, “Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm”, Appl Soft Comput, vol. 11, no. 8, pp. 5782-5792, 2011. https://doi.org/10.1016/j.asoc.2011.01.046

[17] M. Kurdi, ”An effective new island model genetic algorithm for job shop scheduling problem”, Comput Oper Res, vol. 67, pp. 132-142, 2016. https://doi.org/10.1016/j.cor.2015.10.005

[18] R. Cheng, M. Gen, Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, Part II: Hybrid genetic search strategies", Comput Ind Eng, vol. 36, no. 2, pp. 343-364,1999. https://doi.org/10.1016/S0360-8352(99)00136-9

[19] T .Yamada, R. Nakano, “A genetic algorithm applicable to large-scale job-shop problems”. *In Proc*. the 2nd international workshop on parallel problem solving from nature, Brussels, Belgium, 1992, pp. 281–290.

[20] H. P. Lee, S. Salim, “A modified Giffler and Thompson genetic algorithm on the job shop scheduling problem”, MATEMATIKA, 2006, vol. 22, no. 2, pp. 91-107, 2006. https://doi.org/10.11113/matematika.v22.n.178

[21] M. Moonen, G. Janssens, “A Giffler-Thompson Focused Genetic Algorithm for the Static Job-Shop Scheduling Problem”, Journal of Information and Computational Science, vol. 4, no. 2, pp. 629-642, 2007. http://hdl.handle.net/1942/10029

[22] F .Werner, ” A survey of genetic algorithms for shop scheduling problems,. P. Siarry: Heuristics: Theory and Applications, Nova Science Publishers, (2013), pp. 161-222.

[23] M. Kurdi, “An improved island model memetic algorithm with a new cooperation phase for multi-objective job shop scheduling problem”, Comput Ind Eng, vol. 111, pp.183-201, 2007. https://doi.org/10.1016/j.cie.2017.07.021

[24] T. Yamada, “Studies on metaheuristics for jobshop and flowshop scheduling problems”, doctoral dissertation, Kyoto University, Japan, 2003.

[25] A.P. Engelbrecht, “ Computational intelligence: an introduction (Second Edition)”, John Wiley & Sons, West Sussex, England, 2007.

[26] A. Hertz, D. Kobler, “A framework for the description of evolutionary algorithms”, Eur J Oper Res, vol. 126, no. 1, pp. 1-12, 2000. https://doi.org/10.1016/S0377-2217(99)00435-X

[27] L. Gao,G. Zhang, L. Zhang, , X. Li, "An efficient memetic algorithm for solving the job shop scheduling problem", Comput Ind Eng, vol. 60, no. 4, pp. 699-705, 2011. https://doi.org/10.1016/j.cie.2011.01.003

[28] L. Davis, “Job shop scheduling with genetic algorithms”, *In Proc*. An international conference on genetic algorithms and their applications, Carnegie-Mellon University, Pittsburgh, PA, USA, 1985, pp. 136-140

[29] D. Goldberg, “Genetic algorithms in search, optimization and machine learning”, Addison-Wesley, MA, 1989.

[30] M. Gen, R. Cheng, “Genetic algorithms and engineering design, John Wiley & Sons, New York, 1997.

[31] H. Fisher, G.L. Thompson,” Probabilistic learning combinations of local job-shop scheduling rules”, J. Muth, G. Thompson (Eds.), Industrial scheduling, Prentice-Hall, Englewood Cliffs, NJ, 1963, pp. 225-251.

[32] S. Lawrence, “Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques“ (supplement), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.

[33] D. Applegate, W. Cook, “A computational study of the job shop scheduling problem”, ORSA J Comput, vol. 3, no. 2, pp. 149-156, 1991. https://doi.org/10.1287/ijoc.3.2.149

[34] L. Asadzadeh, K. Zamanifar, “ An agent-based parallel approach for the job shop scheduling problem”, Math Comput Model, vol. 52, no. (11–12), pp.1957-1965,2010. https://doi.org/10.1016/j.mcm.2010.04.019

[35] M. Seo, D. Kim, “Ant colony optimisation with parameterised search space for the job shop scheduling problem”, Int J Prod Res, vol. 48, no. 4, pp. 1143-1154, 2010. https://doi.org/10.1080/00207540802538021

[36] G.G. Yen, B. Ivers, “Job shop scheduling optimization through multiple independent particle swarms”, Int J Intell Comput Cybern, vol. 2, no. 1, pp. 5-33, 2009. https://doi.org/10.1108/17563780910939237