Soft Computing (2019) 23:5099–5116
https://doi.org/10.1007/s00500-018-3177-y

METHODOLOGIES AND APPLICATION

Extended Genetic Algorithm for solving open-shop scheduling problem

Ali Asghar Rahmani Hosseinabadi[1] · Javad Vahidi[2] · Behzad Saemi[3] · Arun Kumar Sangaiah[4] · Mohamed Elhoseny[5]

Published online: 7 April 2018

Abstract
Open-shop scheduling problem (OSSP) is a well-known topic with vast industrial applications which belongs to one of the most important issues in the field of engineering. OSSP is a kind of NP problems and has a wider solution space than other basic scheduling problems, i.e., Job-shop and flow-shop scheduling. Due to this fact, this problem has attracted many researchers over the past decades and numerous algorithms have been proposed for that. This paper investigates the effects of crossover and mutation operator selection in Genetic Algorithms (GA) for solving OSSP. The proposed algorithm, which is called *EGA_OS*, is evaluated and compared with other existing algorithms. Computational results show that selection of genetic operation type has a great influence on the quality of solutions, and the proposed algorithm could generate better solutions compared to other developed algorithms in terms of computational times and objective values.

Keywords Extended Genetic Algorithm · Makespan · Crossover · Mutation · Open-shop scheduling

1 Introduction

In open-shop scheduling problem (OSSP), n jobs are processed by m machines. If all of the jobs are processed in a constant route by m machines (e.g., all jobs are processed by machine 1 at first, then by machine 2, and so on until they are processed by machine m), the problem will be a flow-shop type problem. If each job is processed in a specific route by m machines, the problem will be a job-shop type problem; and if the routes of job processing are not deterministic, the problem will be classified as open-shop type (Shamshirband et al. 2015a; Hosseinabadi et al. 2015; Shojafar et al. 2016).

In OSSP, m machines process a set of jobs. Each job includes n operations that must be processed in a deterministic processing time interval by a machine (Hosseinabadi et al. 2013). Operations can be processed through following any possible order. At a specified time, each machine is able to process only one operation of a job. Operations of each job can be processed only after a time called release time for each job. During the process of an operation, no interrupt or tardiness is allowed; and the jobs are independent. The goal of this problem is to find an optimal scheduling with makespan $C_{\max}$ for the completion of n jobs. To achieve this result, the jobs processing sequence and the routes that jobs

Communicated by V. Loia.

✉ Arun Kumar Sangaiah
Arunkumarsangaiah@gmail.com

Ali Asghar Rahmani Hosseinabadi
A.R.Hosseinabadi@iaubs.ac.ir

Javad Vahidi
Jvahidi@iust.ac.ir

Behzad Saemi
Behzad.Saemi@hotmail.com

Mohamed Elhoseny
Mohamed_elhoseny@mans.edu.eg

1 Young Researchers and Elite Club, Ayatollah Amoli Branch, Islamic Azad University, Amol, Iran

2 Iran University of Science and Technology, Tehran, Iran

3 Computer Department, Kavosh Institute of Higher Education, Mahmood Abad, Mazandaran, Iran

4 School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore 632014, India

5 Faculty of Computers and Information, Mansoura University, Mansoura, Egypt

pass through must be considered (Tavakkolai et al. 2015; Farahabadi and Hosseinabadi 2013).

For simplicity, it is assumed that all jobs in most problems are accessible at zero time. According to the scheduling standard introduced by Graham et al. (1979), the problem can be defined in the form of $O_{\max}||C_{\max}$, where m is the number of machines. Pinedo (1995) proposed a prioritization rule in the form of the Longest Alternate Processing Time First (LAPT) for the $O_2||C_{\max}$ problem by which optimal scheduling was obtained in a polynomial time complexity. Gonzalez and Sahni (1976) proved $O_3||C_{\max}$ is an NP-hard problem, and, in 1993, Lawler et al. (1993) proved that $O_m\,|3|\,C_{\max}$ is strongly an NP-hard problem, which means that the optimal solution for the problem is not obtained in a polynomial time complexity. Branch and bound algorithms were used to solve small problems, and heuristic algorithms, which could find approximately optimal solutions, are suitable for solving large problems (Vasant et al. 2016; Vasant 2014).

Broadly speaking, the optimization algorithms can be divided into two categories of exact and approximate methods (Talbi 2009).

In this paper, a new extended GA is proposed to solve the OSSP with the aim of minimizing the end time of all jobs. Also, various efficient crossover and mutation operators are studied in the proposed algorithm. Jobs and operations are as inputs, and the minimization of the finish time of all jobs is the output of the proposed algorithm.

The remaining of the paper is organized as follows: Sect. 2 reviews related works on the subject covered in this paper. Section 3 describes the statement of the problem. Section 4 discusses the proposed algorithm. Sections 5 and 6 present the mechanism of the *EGA_OS* and the computational results, respectively. Finally, the conclusion is presented in Sect. 7.

2 Related work

In this section, some recent works on OSSP are investigated. The parallel OSSP was studied by Chen et al. (2013) with the purpose of minimizing makespan. But due to the NP-hard nature of the problem, it was proposed that approximate algorithms can be used for solving them. Goldansaz et al. (2013) dealt with the multi-processor OSSP with different limitations including independent setup time, process time, and sequence-dependent removal time. It applied the combination of the Imperialist Competitive Algorithm (ICA) and Genetic Algorithm (GA) in order to solve the problem with the objective of minimizing makespan. It has been proved that if restrictions for machine loads were considered, the three-machine proportionate open-shop problem could be solved in $O\,(n \log n)$ time, and if these restrictions were not considered, approximate methods are a suitable alternative for solving them (Koulamas and Kyparisis 2015).

Low and Yeh (2009) first defined the OSSP problem as a binary programming model and then introduced a Hybrid Genetic Algorithm (HGA) considering various restrictions including independent setup and dependent removal time for solving the problem with the purpose of makespan reduction. Other researchers (Shamshirband et al. 2015a) studied a special type of the open-shop problem called no-wait open-shop with the purpose of reducing the makespan. They then used three mathematical models in the form of integer linear programming models and patterns for coding and decoding. Furthermore, these patterns were used to introduce a method based on GA and variable neighborhood search for optimally solving the problem. The Lagrange expansion was used to total quadratic completion time reduction in scheduling small open-shops (Zhang and Bai 2014).

An efficient HGA called Hybrid Non-dominated Sorting Genetic Algorithm (HNSGA) was developed to solve the multi-objective open-shop problem in Noori-Darvish and Tavakkoli-Moghaddam (2011). This method applied the Local Search Algorithm (LSA) in order to generate an initial population for solving medium- and large-sized problems. Results obtained after the implementation with respect to qualitative criteria, variety, and space were found to be superior to the Strength Pareto Evolutionary Algorithm (SPEA-II) method.

Several two-phase heuristic algorithms were applied for solving the OSSP with portable dedicated machines and without time limitations in Hosseinabadi et al. (2014), Hosseinabadi et al. (2012) and Rostami et al. (2015). Furthermore, another HGA with special operations (Ahmdizar and Hosseinabadi 2012) was proposed for solving the OSSP. The suggested operations could omit the sequence of performing the jobs by the machines, and the proposed mutation operation could prevent search process for redundant solutions. The proposed method was a combination of iterative randomized active scheduling concepts, a dispatching index, and a lower bound.

In Ciro et al. (2016), two multi-objective methods named Non-dominated Sorting Genetic Algorithm (NSGA-II) and NSGA-III were proposed and compared. Different constraints, such as tools allocation and multi-skills staff assignment, were considered in these methods. The objectives were to minimize total flow time of jobs in the production system, workload balancing in both manpower and machine points of view. They generated some small- and large-sized instances to show the general performance of these algorithms. Results showed that when the size of instances increases, the performance of NSGA-III was better than that of NSGA-II.

The asymptotic optimality of the General Dense Scheduling (GDS) algorithm was investigated for the static and dynamic versions of the Flexible Open-Shop (FOS) makespan problem in Baia et al. (2016). A GDS-based algorithm was

utilized to improve the original GDS algorithm for large- and average-scale problems.

A mechanical workshop-based OSSP was discussed in Ciro et al. (2015). The problem was formulated as a fuzzy Mixed-Integer Linear Programming (MILP) model and solved using an Ant Colony Optimization (ACO).

A recent study in Azadeh et al. (2016) has developed a novel bi-objective MILP model for multi-objective OSSP. Independent setup time and sequence-dependent transportation time were the main constraints considered in the model. The aim of the proposed model was to minimize the total makespan of jobs as well as the workload of all machines.

Harmanani and Ghosn (2016) proposed a Simulated Annealing Algorithm (SA) for solving the non-preemptive open-shop scheduling problem in order to minimize the makespan by determining a schedule for operations on the machines. Makespan was defined as the time starts from the beginning of the first operation until the end of the last operation.

Other recent studies (Karagöz and Yıldız 2017; Hosseinabadi et al. 2016a, 2017b; Shamshirband et al. 2015b; Hosseinabadi et al. 2016b, 2017a) evaluated the performances of some recent optimization algorithms, such as Particle Swarm Algorithm (PSO), Cuckoo Search Algorithm (CSA), Gravitational Search Algorithm (GSA), Hybrid Gravitational Search-Nelder Mead Algorithm (HGSANM), League Championship Algorithm (LCA), Firefly Algorithm (FA), Bat Algorithm (BA), Interior Search Algorithm (ISA), ICA, Gravitational Emulation Local Search Algorithm (GELS), and SA in terms of finding the optimal thin-walled tube design and Vehicle Routing Problem (VRP).

Yildiz (2012) investigated some population-based optimization methods to solve multi-pass turning optimization problems. The outcome of the investigation was a proposed differential evolution algorithm based on a hybrid technique for solving manufacturing optimization problems.

Another study presented in Yıldız et al. (2007) aimed to develop a more efficient shape optimization approach using GA and robustness issues to release the restrictions caused by a larger population of solutions in the pure Multi-Objective Genetic Algorithm (MOGA).

Recently, Tellache and Boudhar (2017) have introduced open shops considering some conflicting jobs that cannot be processed simultaneously on different machines. They proposed heuristics and lower bounds for the general cases of the problem.

3 Problem description

OSSP is considered as a kind of general job scheduling problem and consists a set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and must be processed on one of the m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job $J_i \in \mathcal{J}$ consists m operations $o_{i,j}$ that is processed on the machine M_i. At any moment, each machine can only process one operation related to the job. Two or more operations of a job cannot be processed on different machines at the same time. The problem is to find an optimal schedule for the operations on machines that minimizes makespan ($C_{\max}$); makespan is the calculated from the beginning time of the first operation until the last one (Bai and Tang 2013). A schedule of optimum finishing time is a schedule containing at least the finishing time among all schedules. If there is at least a triad $<i, s(i), f(i)>$ for each machine and each job, which must be scheduled, then the schedule is non-preemptive. $S(i)$ denotes the ith scheduling, and $f(i)$ denotes the objective function value of ith scheduling. A preemptive schedule is the one that has no constraint for the number of triads for jobs and machines. It has been demonstrated that non-preemptive open-shop scheduling problem is an NP-hard problem because it converts the problem into a scheduling problem (Gonzalez and Sahni 1976). It is proved that the following lower bound is correct for scheduling with optimum finishing time (Gonzalez and Sahni 1976).

$$\mathcal{L} = \left\{ \max_i \sum_{j=1}^{m} p_{i,j}, \max_j \sum_{i=1}^{n} p_{i,j} \right\}, \quad \text{where } 1 \le i \le n \text{ and } 1 \le j \le m. \tag{1}$$

The first part of Eq. (1) is related to the maximum completion time of all jobs that must be processed on machines. On the other hand, it indicates that the total processing time of jobs must at least be equal to their defined required time to process their operations. The second part shows the maximum completion time of jobs assigned to a given machine. Therefore, the processing time of each machine must at least be equal to the sum of all times required for operating its assigned jobs. The optimal makespan of an open-shop scheduling is never less than $\mathcal{L}$, but it does not indicate that it should be necessarily equal to $\mathcal{L}$ (Harmanani and Ghosn 2016).

This research evaluates the OSSP using the 4×4 Taillard Benchmark (Taillard 1993) as shown in Table 1. The benchmark instance consists of 4 jobs and 4 machines. Figure 1 shows a possible schedule with an optimal makespan of 193.

Table 1 A 4×4 *Taillard Benchmark* instance for the OSSP (Taillard 1993)

Jobs	(Processing time, machine)			
Job 1	(54, M_3)	(34, M_1)	(61, M_4)	(2, M_2)
Job 2	(9, M_4)	(15, M_1)	(89, M_2)	(70, M_3)
Job 3	(38, M_1)	(19, M_2)	(28, M_3)	(87, M_4)
Job 4	(95, M_1)	(34, M_3)	(7, M_2)	(29, M_4)

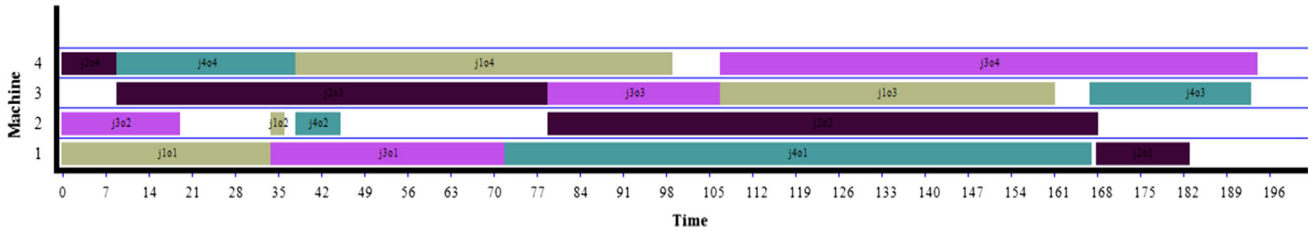

Fig. 1 Optimal schedule for *Taillard Benchmark* instance with the makespan of 193

4 The proposed algorithm

There are many exact methods, heuristics and metaheuristics developed for all introduced optimization problems in order to solve them (Deb and Jain 2014; Khuri and Miryala 1999; Fang et al. 1994; Prins 2000; Colak and Agarwal 2005; Tellache and Boudhar 2017; Tirkolaee et al. 2018, 2017; Mirmohammadi et al. 2017; Babaee Tirkolaee et al. 2016; Alinaghian et al. 2014). In this paper, GA is implemented for developing a novel method in order to solve the OSSP. Also, the effects of the implemented crossover and mutation operations on the proposed approach are investigated to yield the high-quality solutions. Sections 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7 describe the parameters of the proposed algorithm (*EGA_OS*).

4.1 Chromosome representation

The proposed *EGA_OS* applies a single-dimensional array with the same length as the total number of the operations to show each chromosome. Each gene of each chromosome contains a unique integer that indicates one operation in one job. Table 2 shows the procedure of numbering the example system in Table 1.

An example chromosome for the example system in Table 1 is shown in Fig. 2.

As shown in Fig. 2, the first gene is the number 5, which refers to the operation 1 in job number 2 that must be performed by machine 4. In this method of chromosome representation, each chromosome shows a schedule for performing all the operations in each job.

4.2 Fitness function

Makespan is used to determine chromosome fitness and is calculated based on Eq. (2):

$$\text{Fitness} = \text{Max}_1 < I \leq n \{T_i\} \tag{2}$$

where n is the number of jobs and T_i is the completion time for job i.

Table 2 The procedure of numbering the operations for the example system in Table 1

Job	Machine	Operation no.
1	1	2
1	2	4
1	3	1
1	4	3
2	1	2
2	2	3
2	3	4
2	4	1
3	1	1
3	2	2
3	3	3
3	4	4
4	1	1
4	2	3
4	3	2
4	4	4

4.3 Parent selection

The *EGA_OS* used the tournament method to select the parents. In this method, a small subset of the chromosomes is randomly selected and two of the best chromosomes that have the highest fitness values are selected as the parents.

4.4 Crossover operation

Four different crossover operations are used in the *EGA_OS* to study the effects on the type of the selected crossover operation on solving the problem.

4.4.1 Homologous Crossover Genetic Operation (C.S)

Once the parents are selected, the Homologous Crossover Genetic Operation (C.S) first passed the homologous genes in the two parents on to the child, and then the remaining of the genes are stochastically selected from parent 1 or 2 and passed on to the child. In this kind of crossover in the

Fig. 2 Structure of a sample chromosome

5	12	6	8	10	9	3	1	14	7	4	16	13	2	15	11

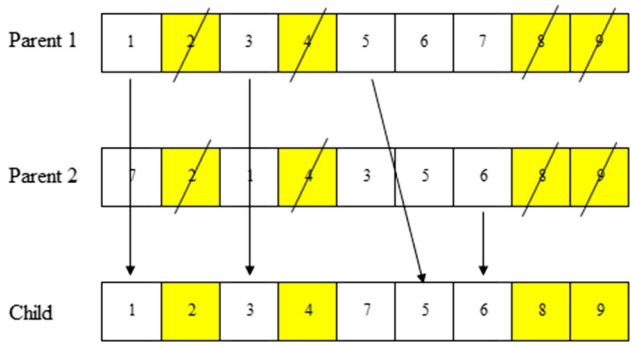

Fig. 3 Homologous Crossover Genetic Operation

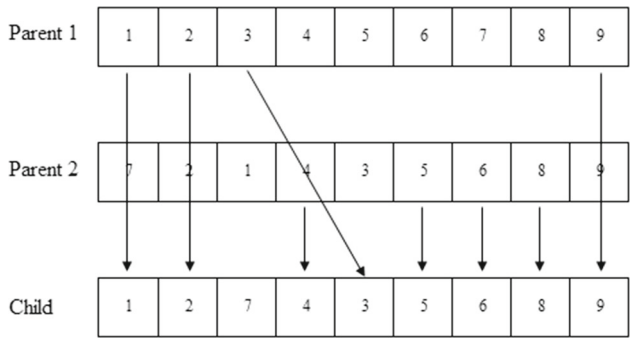

Fig. 4 Row Crossover Operation

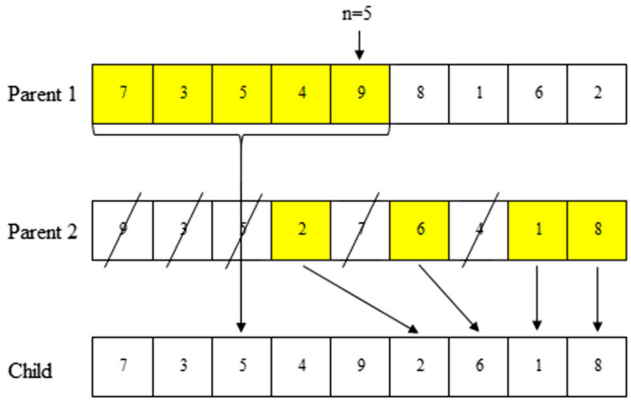

Fig. 5 One-point crossover operation

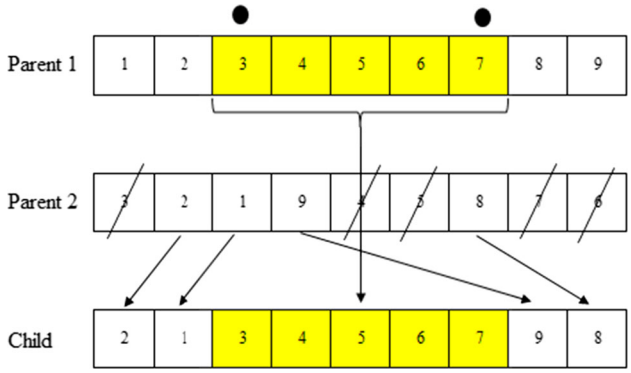

Fig. 6 Two-point crossover operation

early generations, different childrens with greater variety are produced because the chromosomes are different. Therefore, there would be greater diversity in the population. Figure 3 shows the operations performed by the C.S.

4.4.2 Row Crossover Operation (C.R)

Once two parents are selected, the Row Crossover Operation (C.R) randomly selects one of the two parents first and passes its first gene on to the child. This gene is then omitted from the other parent. This process continues until the child received all its genes. In this crossover method, diverse chromosomes are generated. Figure 4 implies an example of the application of the C.R.

4.4.3 One-point crossover operation (C.O)

The One-Point Crossover Operation (C.O) first selects one number stochastically from the interval 1 to n (length of the chromosome) and the genes in the interval from 0 to the randomly selected number are passed on to the child. Homologous genes in parent 2 are then omitted, and the remaining of the genes are embedded, respectively, in the empty cells of the child. Figure 5 illustrates the application of the C.O in the proposed algorithm.

4.4.4 The two-point crossover operation (C.D)

The Two-Point Crossover Operation (C.D) first selects two random numbers in the interval from 0 to n (length of the chromosome). The genes between these two stochastically selected numbers are passed on from parent 1 to the child, and the homologous genes in parent 2 are omitted. The remaining of the genes are embedded, respectively, from parent 2 in the empty cells of the child. Figure 6 implies an example of the application of the C.D.

4.5 Mutation operation

In the *EGA_OS*, two different mutation operations were used to study the effects of the selected type of the mutation operation on solving the problem.

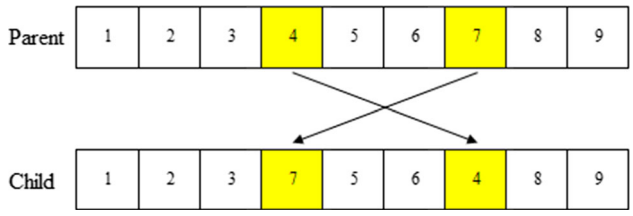

Fig. 7 Displacement Mutation Operation

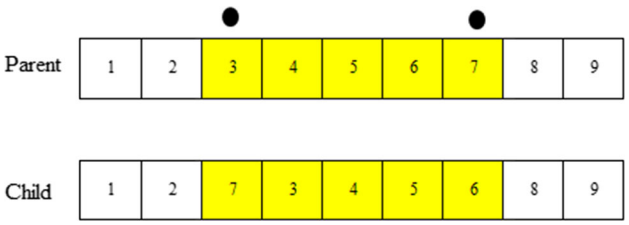

Fig. 8 Shift mutation operation

4.5.1 Displacement Mutation Operation (M.A)

After the selection of the parents, the Displacement Mutation Operation (M.A) randomly selects two genes and displaces them. Figure 7 indicates the application of the M.A.

4.5.2 Shift mutation operation (M.S)

After the parent chromosome is selected, two points are randomly selected in the interval from 1 to n (length of the chromosome) and the genes located between these two points

```
Function EGA_OS (problem)
Input: Population_size, Problem_size, P_crossover, P_mutattion, StopCondition_Number
Output: S_best
Population←Initalize Population (Population_size, Problem_size);
Evaluate Population (Population);
S_best←Get Best Solution (Population);
int number Repeat=1;
While number Repeat <= Stop Condition_Number do
    int i=0;
    While i < P_crossover do
            Child = Parent_1= Parent_2= ø;
            Select Parents (&Parent_1, &Parent_2, Population, Population_size);
                if Random(100)>30 then
                    Child ←Crossover C.O(Parent_1, Parent_2);
                else
                    Child ←Crossover C.D(Parent_1, Parent_2);
                end
            Evaluate Fitness (Child);
            Population←Insert (Population, Child);
            i++;
        end
        i=0;
        While i< P_mutation do
            Child = Parent_1= ø;
            Select Parents (&Parent_1, Population, Population_size);
                if Random(100)>40 then
                    Child ← Mutate M.A (Parent_1);
                else
                    Child ← Mutate M.S (Parent_1);
                end
            Evaluate Fitness (Child);
            Population←Insert Population (Population, Child);
            i++;
        end
        Population←Select Generation (Population, Population_size);
        S_best←Get Best Solution (Population);
end
Return S_best
```

Fig. 9 The pseudo-code of the *EGA_OS*

Table 3 The parameters of the *EGA_OS*

Population size	Maximum number of generations	Crossover ratio	Mutation ratio
92	250	0.8	0.2

are rotationally shifted to the left. Figure 8 shows an example of M.S.

4.6 Selection of chromosomes for the next generation

Chromosomes for the next generation are selected in the *EGA_OS* using a hybrid approach. The chromosomes are ordered based on their fitness at first, and repetitive chromosomes are then omitted. Only 10% of the chromosomes with higher fitness values are then selected for the next generation. The remaining of the chromosomes are selected at random because this method tried to maintain chromosome dispersion in each iteration.

4.7 Termination condition

Termination condition of GA depends on the type of the problem and the external knowledge about it. In the *EGA_OS*, the specified maximum number of generations is considered as the termination condition, i.e., if the desired number of generations is reached, the best chromosome is found and the algorithm is terminated.

Figure 9 shows the pseudo-code of the *EGA_OS*.

After the initial evaluation of the various mentioned crossover and mutation operations for the OSSP, we reach the conclusion that the mutation and crossover operations substantially influence the achievement of optimal solutions. The conclusion from the initial evaluation is that the One-point and Two-point crossover operations perform better than the other crossover operations. The greater utilization of the mentioned mutation operations enhances scanning capability of the GA in solving the OSSP. Therefore, the *EGA_OS* pseudo-code algorithm is a combination of the two crossover operations (the One-point and Two-point crossover operations), with the C.O having a 70% chance, and the C.D having a 30% chance, of being selected for creating children. Moreover, both displacement and shift mutation operations are selected to apply mutation, with an M.A and M.S having a 60 and 40% chances, respectively, of being selected for creating the children. In the initial evaluations, 60 and 70% probabilities are obtained.

4.8 Parameter setting

The parameters of the *EGA_OS* are determined as below:

Crossover probability is equal to 0.8, and mutation probability is equal to 0.2. The number of generations is 250, and the population size is considered to be 92 according to Deb and Jain (2014). Table 3 shows the parameters of the *EGA_OS*.

5 Evaluation of the proposed algorithm

This section discusses the evaluation of the *EGA_OS* using different crossover and mutation operations to see for which operations the *EGA_OS* provided optimal solutions. This process is described in detail below. The C#.Net programming language was used to implement the algorithms. The algorithms were executed on a computer equipped by Intel ® Pentium ® 4 CPU 3.00 GHz processor and 2.00GB of RAM. In order to investigate the efficiency of the *EGA_OS* and the effects of crossover and mutation operations on solving the OSSP, 9 datasets in three different groups were designed.

The designed datasets were named *Test_N_M*, where *N* is the number of jobs and *M* is the number of operations in each job of the designed datasets. Table 4 shows the designed datasets.

Table 4 The designed test datasets

Groups of datasets	Test	Number of jobs	Number of operations	Initial population	Number of generations
Group 1 test datasets	Test_4_4	4	4	100	500
	Test_5_5	5	5	100	500
	Test_6_6	6	6	100	1000
Group 2 test datasets	Test_7_7	7	7	150	20,000
	Test_8_8	8	8	150	3000
	Test_10_10	10	10	150	4000
Group 3 test datasets	Test_15_15	15	15	200	5000
	Test_20_20	20	20	250	6000
	Test_30_20	30	20	250	7000

Table 5 Results of executing the *EGA_OS* on group one test datasets

Test_N_M	LB	UB	CPU_s	MU	CR
Test_4_4	**287**	774	18	MA	C.O
	287	775	17		C.D
	287	826	19		C.S
	287	868	20		C.R
Test_4_4	**287**	723	17	M.S	C.O
	295	750	18		C.D
	317	781	18		C.S
	313	815	20		C.R
Test_5_5	**350**	838	18	M.A	C.O
	371	918	18		C.D
	356	1034	22		C.S
	356	1048	22		C.R
Test_5_5	364	796	19	M.S	C.O
	350	899	20		C.D
	400	1083	21		C.S
	389	964	23		C.R
Test_6_6	**436**	1232	40	M.A	C.O
	439	1254	39		C.D
	488	1396	47		C.S
	487	1365	50		C.R
Test_6_6	477	1254	41	M.S	C.O
	445	1262	42		C.D
	497	1643	45		C.S
	486	1638	51		C.R

Bold indicates the highest fitness value

Table 6 Results of executing the *EGA_OS* on group 2 test datasets

Test_N_M	LB	UB	CPU_s	MU	CR
Test_7_7	**522**	1513	123	MA	C.O
	559	1560	124		C.D
	552	1841	160		C.S
	536	1758	177		C.R
Test_7_7	654	1543	115	M.S	C.O
	552	1529	123		C.D
	625	1794	123		C.S
	588	1769	175		C.R
	578	1584	202		C.O
	578	1625	202		C.D
	578	2074	272		C.S
	633	2138	302		C.R
Test_8_8	613	1543	197	M.S	C.O
	582	1707	198		C.D
	618	1959	216		C.S
	608	2002	298		C.R
Test_10_10	**635**	2085	336	M.A	C.O
	672	2063	324		C.D
	683	2527	476		C.S
	661	2481	543		C.R
Test_10_10	**652**	1987	333	M.S	C.O
	653	2041	330		C.D
	709	2537	381		C.S
	759	2437	535		C.R

Bold indicates the highest fitness value

Table 5 shows the results of the execution of the *EGA_OS* on the test datasets in group 1, where *LB* shows the highest fitness value, *UB* is the lowest fitness value, *CPUs* is the time (in seconds) required for executing the algorithm, and *MU* and *CR* are the type of mutation and crossover operation, respectively, employed in the *EGA_OS*. As shown in Table 5, in group 1 dataset the use of the C.O yielded better solutions compared to the other crossover operations. The M.S Mutation Operation together with the C.D performed relatively better compared to the other operations.

Table 6 lists results of executing the *EGA_OS* on group 2 test datasets. Here too, the use of the C.O yielded better solutions, and the M.A performed relatively better compared to the M.S.

Table 7 illustrates the results of executing the *EGA_OS* on group 3 test datasets. As observed in Table 7, once again the use of the C.O and the M.A was more suitable and yielded better solutions, whereas the M.S mutation operation together with the C.O crossover operation performed poorly. Moreover, the C.D performed better than the C.R and the C.S.

Figure 10 shows the fitness diagram of the *EGA_OS* for the Test_7_7 dataset based on different crossover and mutation operations (M.A).

Figures 11 and 12 show the dispersion diagrams for the Test_8_8 dataset and Test_5_5, respectively. Population dispersion is always maintained in the *EGA_OS*. This is one of the reasons for using the hybrid approach that prevented premature convergence of the chromosomes. Moreover, comparison of the dispersion diagrams of the various crossover operations in Figs. 11 and 12 implies the C.R and C.S crossover operations searched greater space of the problem compared to the other crossover operations reason being they generated more diverse chromosomes compared to the other crossover operations but did not find better solutions than the C.O crossover operation. The C.O crossover operation generated more desirable chromosomes, and better solutions compared to the other crossover operations were obtained because it merged information that resulted from the two parents better.

Figure 13 shows fitness comparison on Test_15_15 using various crossover and mutation operations. As shown in Fig. 13 the M.A mutation operation was more efficient com-

Table 7 Results of executing the *EGA_OS* on group 3 test datasets

Test_N_M	LB	UB	CPU_s	MU	CR
Test_15_15	**1256**	3923	999	MA	C.O
	1303	4253	1034		C.D
	1432	5585	1470		C.S
	1422	6051	1802		C.R
Test_15_15	**1358**	3901	1047	M.S	C.O
	1403	4032	978		C.D
	1708	5768	1209		C.S
	1695	6043	1916		C.R
	1694	4396	2801		C.O
	1766	4790	2685		C.D
	1909	6755	3014		C.S
	1900	5064	4119		C.R
Test_20_20	**1752**	4458	2790	M.S	C.O
	1753	4806	2559		C.D
	2141	8340	3231		C.S
	2213	7449	5922		C.R
Test_30_20	2470	9483	7453	M.A	C.O
	2499	9565	7739		C.D
	2922	11926	8577		C.S
	2453	11629	12481		C.R
Test_30_20	2706	9483	6043	M.S	C.O
	2466	9483	9390		C.D
	3413	11926	13424		C.S
	3601	12121	20718		C.R

Bold indicates the highest fitness value

pared to the M.S mutation operation. Moreover, Fig. 13 indicates better solutions could be obtained by using the M.A mutation operation together with the C.O crossover operation. By applying the M.A mutation operation, the generated children are very similar to the parents. These children differed in only two genes, and this similarity to the parents gave better parents a greater opportunity to create children, which made it easier to obtain the solution in the problem space. However, if the M.S mutation operation was used, children very different from the parents could be created. This would cause greater diversity in chromosomes, but, compared to the M.A mutation operation, the solution would be found much more slowly.

Figure 14 shows the Gantt chart of an example of open-shop scheduling for the Test_6_6 dataset in which the C.O and the M.A were used for solving the problem. Figure 15 shows the Gant chart of an example of open-shop scheduling of the Test_7_7 dataset.

6 Computational results for the designed test problems

For this study, 20 test datasets were considered to investigate the efficiency of the *EGA_OS* and the effects of the selection of crossover and mutation operators on solving the OSSP. The test datasets are named as *Test_N_M*, where *N* was the number of jobs and *M* was the number of operations of each job in the dataset. The designed test datasets were divided into three groups. The first group represented small systems,

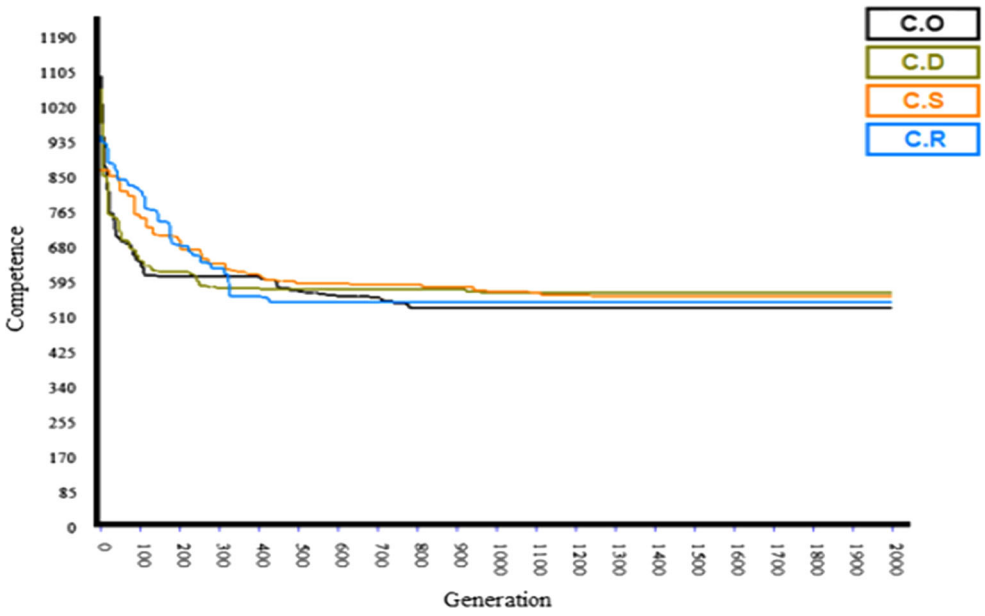

Fig. 10 Fitness diagram of the *EGA_OS* for the Test_7_7 dataset, based on various crossover and mutation operations (M.A)

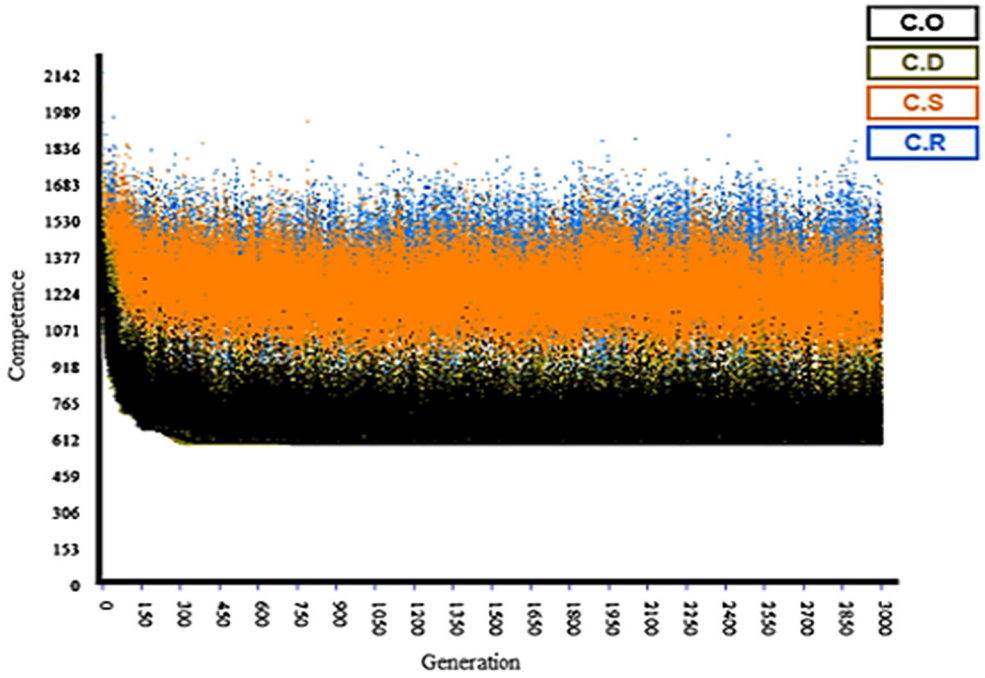

Fig. 11 Dispersion diagram of the various crossover operations for the Test_8_8 dataset

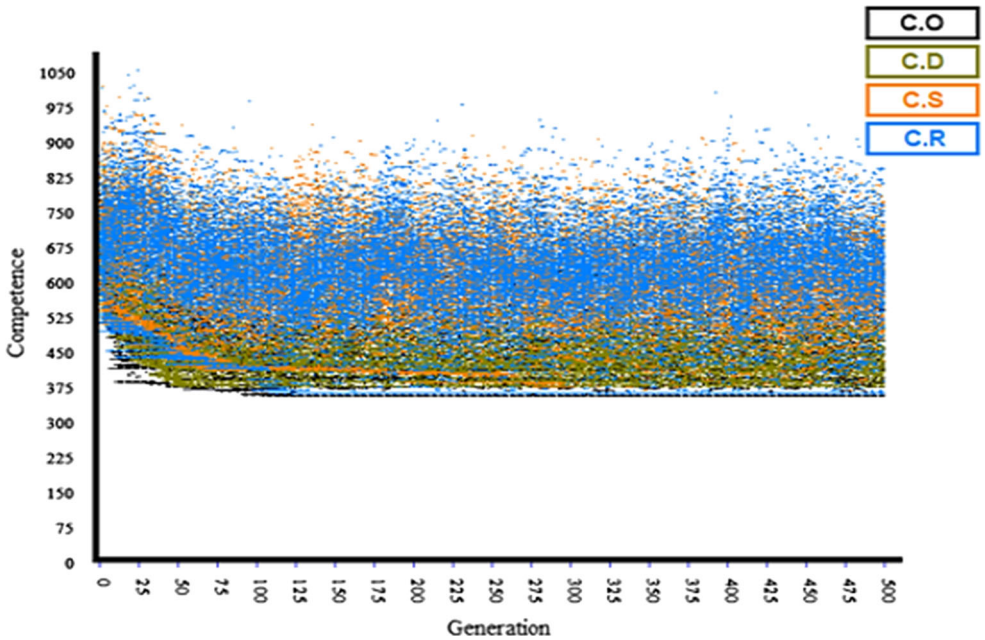

Fig. 12 Dispersion diagram of the various crossover operations for the Test_5_5 dataset

the second group indicated medium systems, and the third group represented large systems.

Table 8 presents results of executing the *EGA_OS* on the test datasets of small, medium, and large systems and compares these results with those of the ACO and ICA Algorithm. In Table 8, *LB* represents the highest fitness value, *UB* the lowest fitness value, and *CPUs* the time (in seconds) required for executing the algorithm. As shown in Table 8, the *EGA_OS* could find more optimal solutions for all kinds of problems compared to the other two algorithms and could be executed in a shorter time.

Figures 16, 17, 18, 19, 20, and 21 compare the fitness of the *EGA_OS* and those of ACO and the ICA algorithms for some of the designed datasets. As evident from Figs. 16, 17, 18, 19, 20, and 21, *EGA_OS* could solve the problem considered for this study efficiently and compete with compared algorithms.

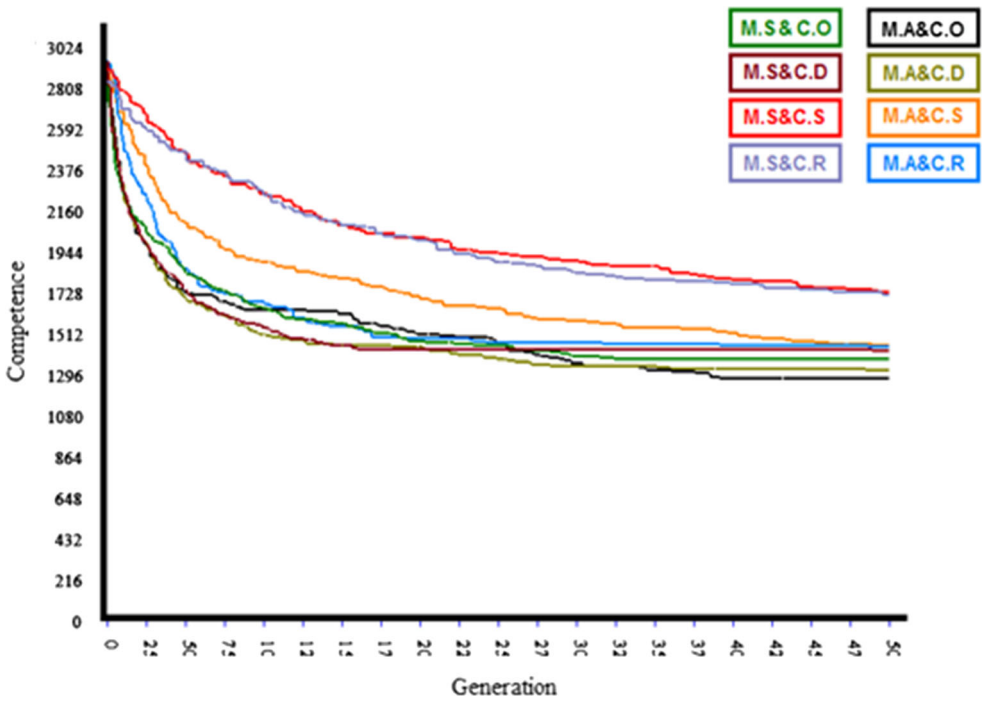

Fig. 13 Fitness diagram of various crossover and mutation operations for the Test_15_15 dataset

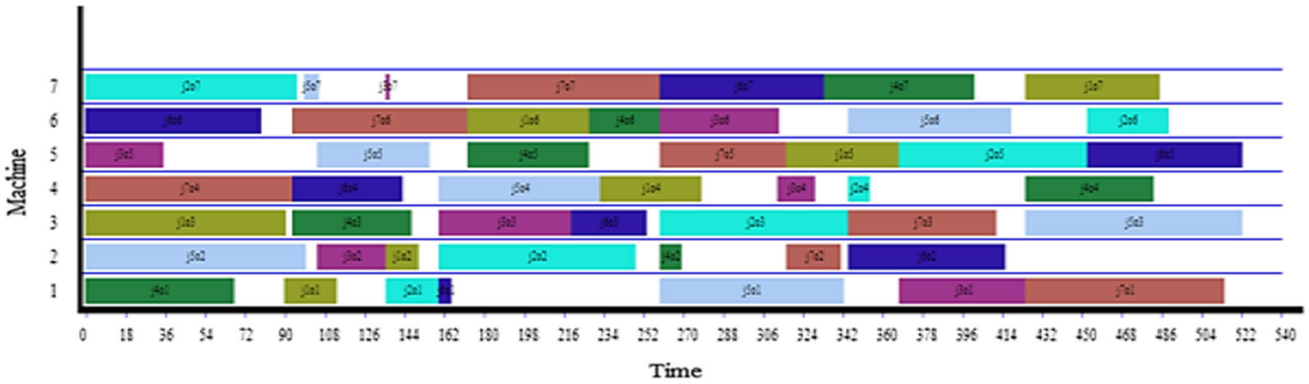

Fig. 14 Gantt chart of an example of open-shop scheduling for the Test_6_6 dataset

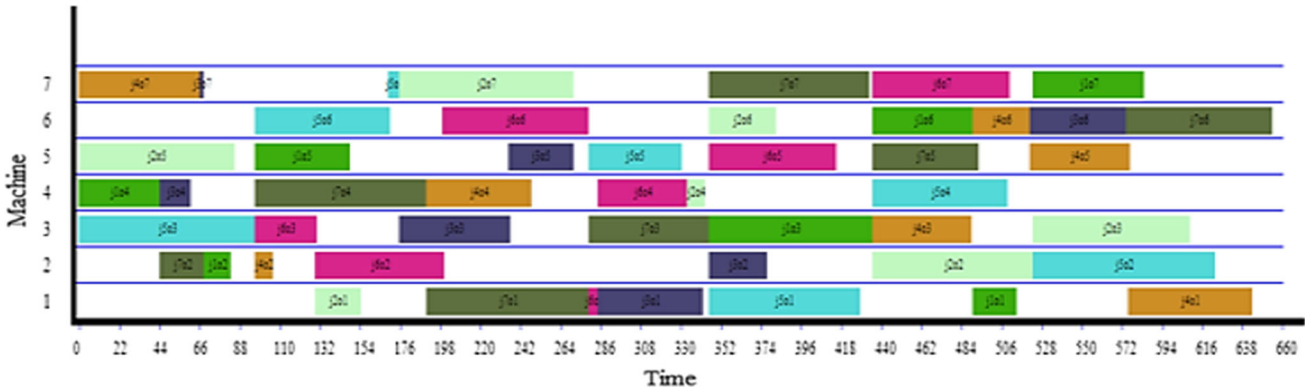

Fig. 15 Gantt chart of an example of open-shop scheduling for the Test_7_7 dataset

Table 8 Results of executing the *EGA_OS, ACO*, and *ICA* algorithms on various test data

Types of test data	Problem (Test_N_M)	EGA_OS			ACO			ICA		
		LB	UB	CPUs	LB	UB	CPUs	LB	UB	CPUs
Small test data set	Test_2_2	132	215	6	132	215	4	132	215	6
	Test_3_3	225	373	7	225	422	5	225	489	8
	Test_4_4	287	669	18	318	733	15	306	788	19
	Test_5_5	350	904	39	354	828	29	361	871	38
	Test_6_6	471	1301	68	479	1297	53	473	1297	63
	Test_7_7	514	1409	108	570	1760	92	566	1541	100
Medium test data sets	Test_8_8	578	1472	153	578	1624	153	578	1624	154
	Test_9_9	604	1863	188	647	1942	211	640	1979	223
	Test_10_10	628	2015	266	705	2015	224	681	2060	247
	Test_12_12	979	2751	632	957	3094	238	972	3068	248
	Test_15_15	1256	3906	1017	1273	3906	937	1393	3906	1027
	Test_18_18	1512	5088	1160	1568	5088	1049	1526	5088	1172
	Test_20_20	1774	4597	1524	1776	4432	1275	1789	4342	1453
Large test data sets	Test_25_25	2468	5681	2809	2544	5468	2300	2479	5513	2702
	Test_30_20	2445	9483	5871	2448	9483	3143	2538	9483	5579
	Test_30_25	2768	11,055	6981	2944	11,098	4609	2807	11,212	8150
	Test_30_30	3134	10,555	8263	3213	1207	6386	3255	12,017	12,414
	Test_40_20	3064	7722	6395	3067	7113	6277	3112	7694	9213
	Tesr_40_25	3447	7911	9449	4458	8595	7310	3767	8686	12,808
	Test_50_30	7342	19,653	50,050	8242	19,001	44,200	8765	20,155	52,650

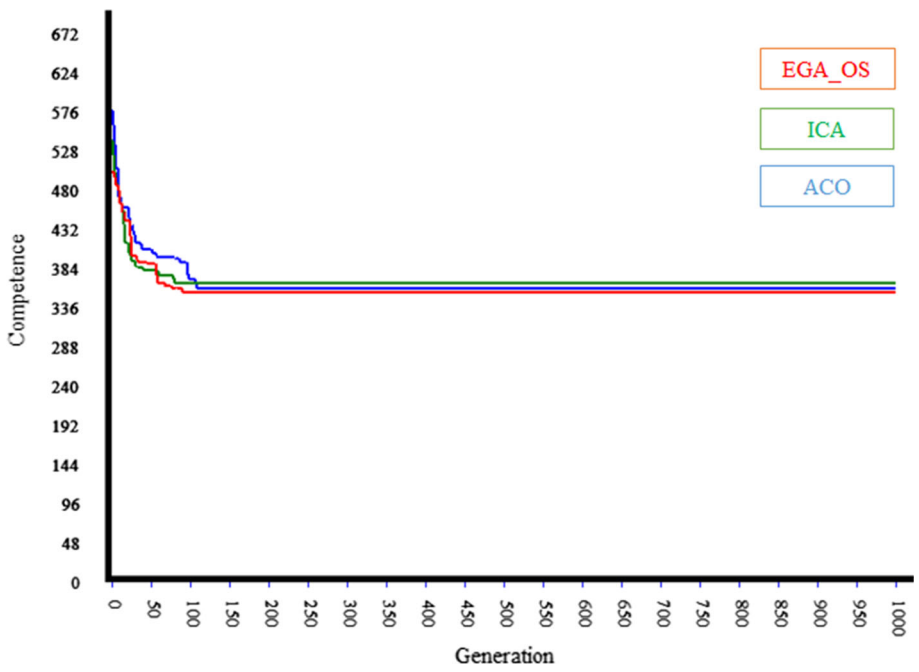

Fig. 16 Fitness diagram of the *EGA_OS* vs *ICA* and *ACO* for the Test_5_5 dataset

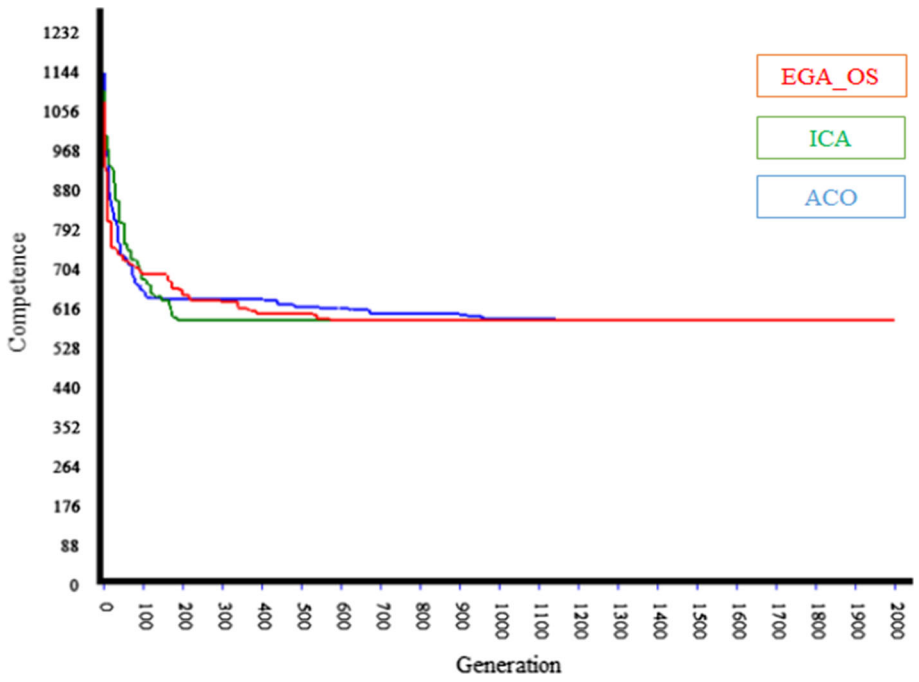

Fig. 17 Fitness diagram of the *EGA_OS* vs *ICA* and *ACO* for the Test_8_8 dataset

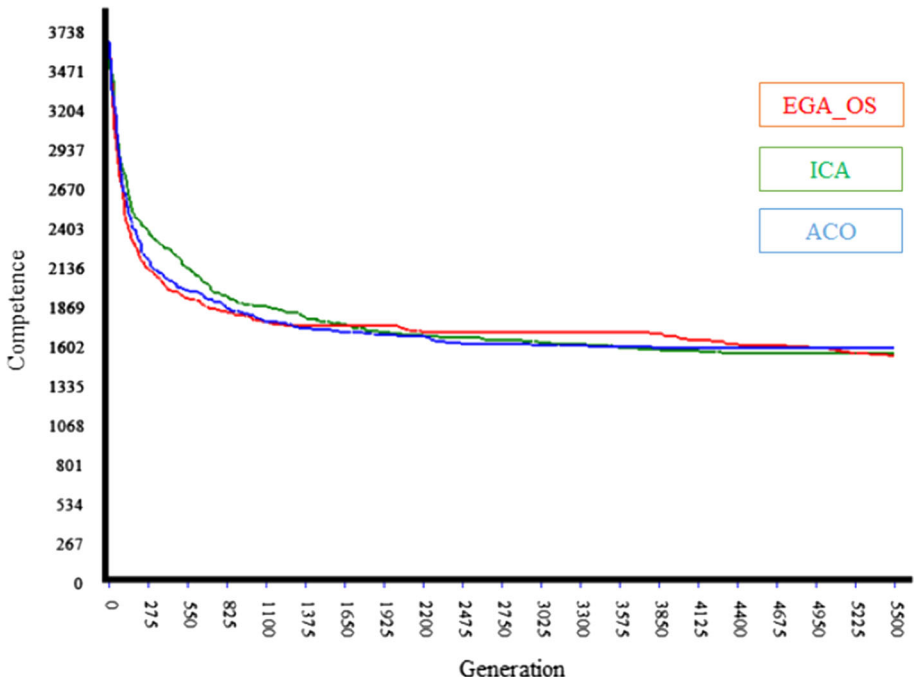

Fig. 18 Fitness diagram of the *EGA_OS* vs *ICA* and *ACO* for the Test_18_18 dataset

The main objective of this assessment was to investigate the convergence of the algorithm in achieving an optimal or a near-optimal solution. In approximation algorithms, population convergence had a direct impact on ending and finding the optimal solution. Generally, some approximation algorithms were converged in a lower number of repetitions and this convergence may occur in an inappropriate space of the problem, so that obtained solution was not optimal. In addition, in some algorithm which emphasizes on population diversity, population convergence occurs slowly and it takes longer time. Thus, an algorithm is useful if it could investigate more space of solutions in a shorter time and also convergence occurs in a short time. Hence, it can generate better solutions than other algorithms.

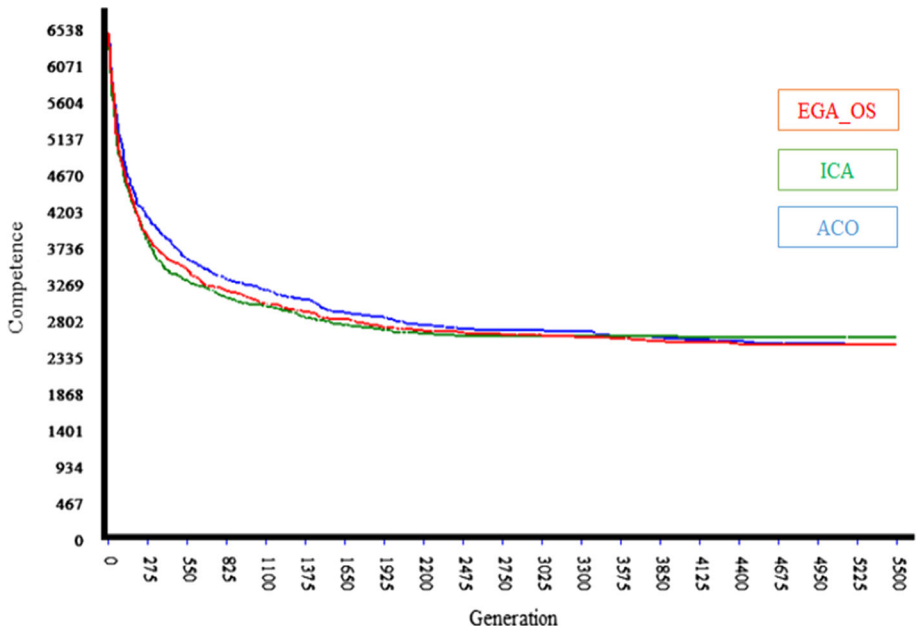

Fig. 19 Fitness diagram of the *EGA_OS* vs *ICA* and *ACO* for the Test_30_20 dataset

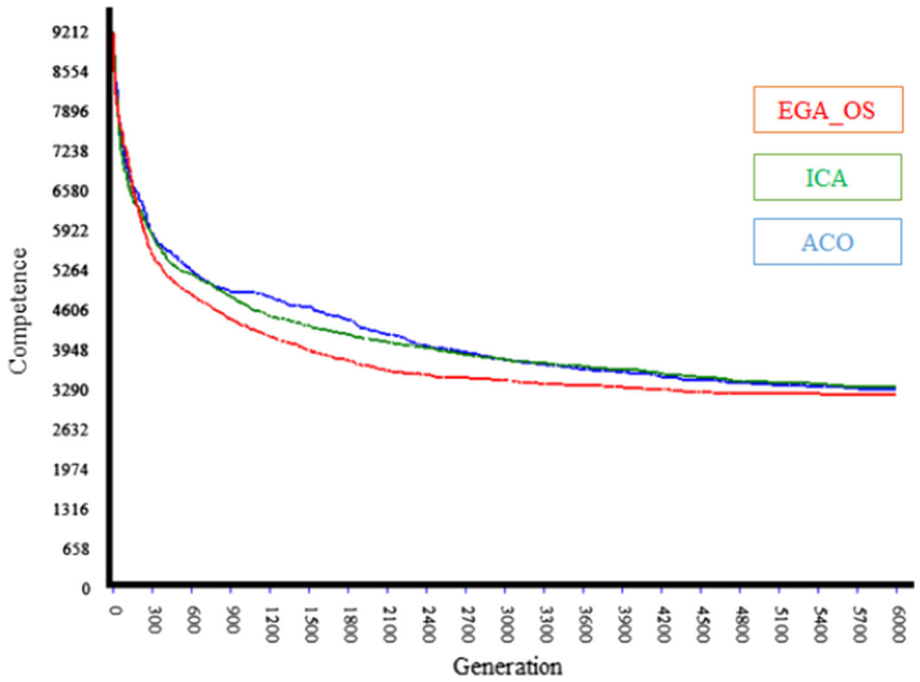

Fig. 20 Fitness diagram of the *EGA_OS* vs *ICA* and *ACO* for the Tset_30_30 dataset

Generally, solutions' diversity has a direct impact on finding the optimal solution. If an algorithm investigates more space of problem in a shorter time, and it gets more time to converge, the chance of achieving optimum solution increases.

7 Computational results for the benchmarks

This section provides some scenarios containing different jobs and machines for solving the problem considered for this study using the *EGA_OS*. The created scenarios were divided into three groups according to the number of jobs and scenario complexity. Table 9 shows these scenarios with their number of jobs and machines.

The *EGA_OS* was tested on Taillard Benchmarks (Taillard 1993). Tables 10 and 11 describe the obtained results of Taillard Benchmark using determined parameters. As it can be seen, the *EGA_OS* could obtain optimum solutions in most benchmark instances and compete with compared algorithms. The running time of all benchmark instances is less than five minutes.

Figure 22 shows the best solution for the instance problem of benchmark 10×10^{-2} obtained by the *EGA_OS*.

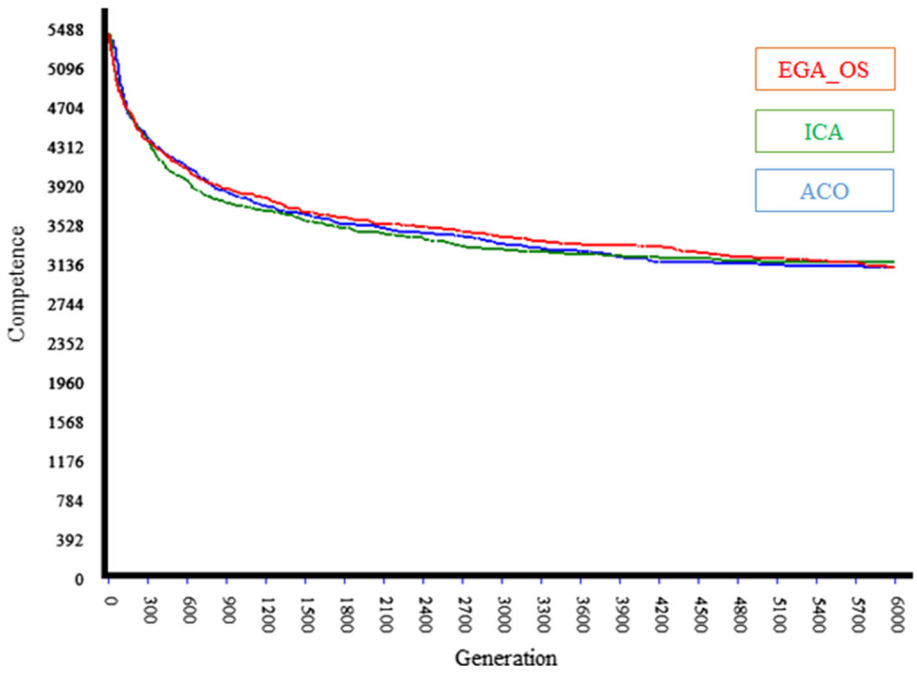

Fig. 21 Fitness diagram of the *EGA_OS* vs *ICA* and *ACO* for the Test_40_20 dataset

Table 9 The classified instances

Classes	Number of jobs	Number of machines	Representation
Small-sized instances	4	4	4×4
	5	5	5×5
Medium-sized instances	7	7	7×7
	10	10	10×10
Large-sized instances	15	15	15×15
	20	20	20×20

8 Conclusion

In this paper, the effects of selected crossover and mutation operators on the proposed algorithm (*EGA_OS*) for solving the Open-Shop Scheduling Problem are investigated and it has been proved that these operators have greatly influenced the efficiency of the Genetic Algorithm. Also, the proposed algorithm is compared with the other algorithms, and the results showed the *EGA_OS* could find more optimal solutions for all kinds of problems and could find them in shorter computational times compared to the other algorithms. In other words, applying the proposed Genetic Algorithm (*EGA_OS*) for solving the Open-Shop Scheduling Problem was largely dependent on selecting the type of

Table 10 Computational results for the Benchmark of Taillard (Taillard 1993) 4×4, 5×5, 7×7, and 10×10

Problem instance	Optimal	SA (Harmanani and Ghosn 2016)	GA (Khuri and Miryala 1999)	Hybrid GA (Khuri and Miryala 1999)	Hybrid GA (Fang et al. 1994)	GA (Prins 2000)	EGA_OS
$4 \times 4 - 1$	193	193	193	213	193	193	193
$4 \times 4 - 2$	236	236	236	240	236	239	236
$4 \times 4 - 3$	271	271	271	293	271	271	271
$4 \times 4 - 4$	250	250	250	253	250	250	250
$4 \times 4 - 5$	295	295	295	303	295	295	295
$4 \times 4 - 6$	189	189	189	209	189	189	189
$4 \times 4 - 7$	201	201	201	203	201	201	201
$4 \times 4 - 8$	217	217	217	224	217	217	217
$4 \times 4 - 9$	261	261	261	281	261	261	261
$4 \times 4 - 10$	217	217	217	230	217	217	217
$5 \times 5 - 1$	300	300	301	323	300	301	300

Table 10 continued

Problem instance	Optimal	SA (Harmanani and Ghosn 2016)	GA (Khuri and Miryala 1999)	Hybrid GA (Khuri and Miryala 1999)	Hybrid GA (Fang et al. 1994)	GA (Prins 2000)	EGA_OS
5 × 5 – 2	262	262	262	269	262	263	262
5 × 5 – 3	323	323	331	353	323	335	323
5 × 5 – 4	310	310	N/A	N/A	310	316	310
5 × 5 – 5	326	326	N/A	N/A	326	330	326
5 × 5 – 6	312	312	312	327	312	312	312
5 × 5 – 7	303	303	N/A	N/A	303	308	303
5 × 5 – 8	300	300	N/A	N/A	300	304	300
5 × 5 – 9	353	353	353	373	353	358	353
5 × 5 – 10	326	326	326	341	326	328	326
7 × 7 – 1	435	435	438	447	435	436	435
7 × 7 – 2	443	443	455	454	443	447	443
7 × 7 – 3	468	468	N/A	N/A	468	472	468
7 × 7 – 4	463	463	N/A	N/A	463	463	463
7 × 7 – 5	416	416	N/A	N/A	416	417	416
7 × 7 – 6	451	451	N/A	N/A	451	455	451
7 × 7 – 7	422	422	443	450	422	426	422
7 × 7 – 8	424	424	N/A	N/A	424	424	424
7 × 7 – 9	458	458	465	467	458	458	458
7 × 7 – 10	398	398	405	406	398	398	398
10 × 10 – 1	637	637	667	655	637	637	637
10 × 10 – 2	588	588	N/A	N/A	588	588	588
10 × 10 – 3	598	598	N/A	N/A	598	598	598
10 × 10 – 4	577	577	586	581	577	577	577
10 × 10 – 5	640	640	N/A	N/A	640	640	640
10 × 10 – 6	538	538	555	541	538	538	538
10 × 10 – 7	616	616	N/A	N/A	616	616	616
10 × 10 – 8	595	595	N/A	N/A	595	595	595
10 × 10 – 9	595	595	627	598	595	595	595
10 × 10 – 10	596	596	623	605	596	596	596

Table 11 Computational results for the Benchmark of Taillard (Taillard 1993) 15 × 15 and 20 × 20

Problem Instance	Optimal	SA (Harmanani and Ghosn 2016)	GA (Khuri and Miryala 1999)	Hybrid GA (Khuri and Miryala 1999	Hybrid GA (Fang et al. 1994)	GA (Prins 2000)	AugNN (Colak and Agarwal 2005)	EGA_OS
15 × 15 – 1	937	937	967	937	937	937	937	937
15 × 15 – 2	918	918	N/A	N/A	918	918	918	918
15 × 15 – 3	871	871	904	871	871	871	871	871
15 × 15 – 4	934	934	969	934	934	934	934	934
15 × 15 – 5	946	946	N/A	N/A	946	946	946	946
15 × 15 – 6	933	933	N/A	N/A	933	933	933	933
15 × 15 – 7	891	891	N/A	N/A	891	891	891	891
15 × 15 – 8	893	893	928	893	893	893	893	893
15 × 15 – 9	899	899	N/A	N/A	899	899	899	899
15 × 15 – 10	902	902	N/A	N/A	902	902	902	902
20 × 20 – 1	1155	1155	1230	1165	1155	1115	1115	1155
20 × 20 – 2	1241	1241	N/A	N/A	1241	1241	1242	1241
20 × 20 – 3	1257	1282	1292	1257	1257	1257	1173	1257
20 × 20 – 4	1248	1274	N/A	N/A	1248	1248	1248	1248
20 × 20 – 5	1256	1289	1315	1256	1256	1256	1256	1256
20 × 20 – 6	1204	1204	1266	1207	1204	1204	1204	1204

Table 11 continued

Problem Instance	Optimal	SA (Harmanani and Ghosn 2016)	GA (Khuri and Miryala 1999)	Hybrid GA (Khuri and Miryala 1999	Hybrid GA (Fang et al. 1994)	GA (Prins 2000)	AugNN (Colak and Agarwal 2005)	EGA_OS
$20 \times 20 - 7$	1294	1294	N/A	N/A	1294	1294	1294	1294
$20 \times 20 - 8$	1169	1169	N/A	N/A	1173	1169	1173	1170
$20 \times 20 - 9$	1289	1307	1339	1289	1289	1289	1289	1289

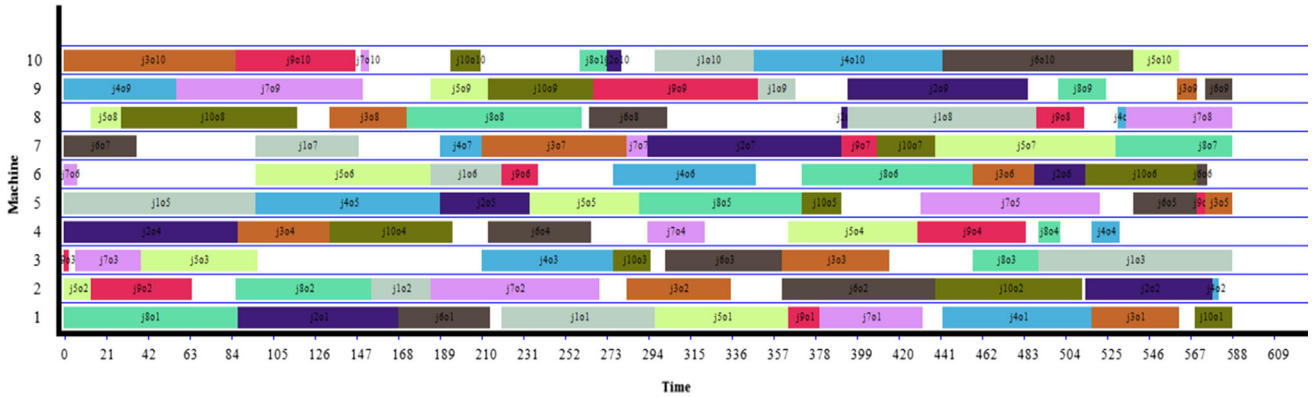

Fig. 22 Schedule for an instance of benchmark 10×10^{-2} with a makespan of 588

crossover and mutation operators. Employment of suitable crossover and mutation operators in the *EGA_OS* led to a goal-oriented dispersion of the chromosomes in the problem space and to find better solutions. Results show that hybrid selection in the Genetic Algorithm worked well for solving the Open-Shop Scheduling Problem and that the use of the One-point Crossover Operator together with the Displacement Mutation Operator resulted in finding solutions better and faster. However, applying different operators resulted in increasing the time complexity of the proposed solution method compared to the state-of-the-art methods. As a future work, it is recommended to make some changes in the proposed method that can parallelly work for different types of mutation and crossover operators. Also, we can consider more real assumptions in the model such as jobs release date and synchronous processing cycles.

Acknowledgements The authors are very thankful for the suggested comments of the Editor in Chief and reviewers to improve our paper.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Ahmdizar F, Hosseinabadi M (2012) A novel hybrid genetic algorithm for the open-shop scheduling problem. Int J Adv Manuf Technol 62:775–787

Alinaghian M, Amanipour H, Tirkolaee EB (2014) Enhancement of inventory management approaches in vehicle routing-cross docking problems. J Supply Chain Manag Syst 3(3):27–34

Azadeh A, Goldansaz SM, Zahedi-Anaraki A (2016) Solving and optimizing a bi-objective open-shop scheduling problem by a modified genetic algorithm. Int J Adv Manuf Technol 85:1603–1613

Babaee Tirkolaee E, Alinaghian M, Bakhshi Sasi M, Seyyed Esfahani MM (2016) Solving a robust capacitated arc routing problem using a hybrid simulated annealing algorithm: a waste collection application. J Ind Eng Manag Stud 3(1):61–76

Bai D, Tang L (2013) Open shop scheduling problem to minimize makespan with release dates. Appl Math Model 37:2008–2015

Baia D, Zhang Z, Zhang Q (2016) Flexible open-shop scheduling problem to minimize makespan. Comput Oper Res 67:207–215

Chen Y, Zhang A, Chen G, Dong J (2013) Approximation algorithms for parallel open-shop scheduling. Inf Process Lett 113:220–224

Ciro GC, Dugardin F, Yalaoui F, Kelly R (2015) A fuzzy ant colony optimization to solve an open-shop scheduling problem with multi-skills resource constraints. In: 8th IFAC conference on manufacturing modelling, management and control MIM 2015, vol 48, pp 715–720

Ciro GC, Dugardin F, Yalaoui F, Kelly R (2016) A NSGA-II and NSGA-III comparison for solving an open-shop scheduling problem with resource constraints. In: 8th IFAC conference on manufacturing modelling, management and control MIM 2016, vol 49, pp 1272–1277

Colak S, Agarwal A (2005) Non-greedy heuristics and augmented neural networks for the open-shop scheduling problem. Naval Res Logist 52:631–644

Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach. IEEE Trans Evol Comput 18:602–622

Fang H-L, Ross P, Corne D (1994) A promising hybrid GA/heuristic approach for open-shop scheduling problems. In: Proceedings of the 11th European conference on artificial intelligence, pp 590–594

Farahabadi AB, Hosseinabadi AR (2013) Present a new hybrid algorithm scheduling flexible manufacturing system consideration cost maintenance. Int J Sci Eng Res 4(9):1870–1875

Goldansaz SM, Jolai F, ZahediAnaraki AH (2013) A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open-shop. Appl Math Model 37:9603–9616

Gonzalez T, Sahni S (1976) Open-shop scheduling to minimize finish time. J Assoc Comput Mach 23:665–679

Graham RL, Lawler EL, Lenstra JK, Kan AHG (1979) Optimization and approximation in deterministic machine scheduling: a survey. Ann Discrete Math 5:287–326

Harmanani HM, Ghosn SB (2016) An efficient method for the open-shop scheduling problem using simulated annealing. Inf Technol New Gener Adv Intell Syst Comput 448:1183–1193

Hosseinabadi AR, Yazdanpanah M, Rostami AS (2012) A new search algorithm for solving symmetric traveling salesman problem based on gravity. World Appl Sci J 16(10):1387–1392

Hosseinabadi AR, Farahabadi AB, Rostami MS, Lateran AF (2013) Presentation of a new and beneficial method through problem solving timing of open-shop by random algorithm gravitational emulation local search. Int J Comput Sci Issues 10(1, No 2):745–752

Hosseinabadi AR, Kardgar M, Shojafar M, Shamshirband S, Abraham A (2014) GELS-GA: hybrid metaheuristic algorithm for solving multiple travelling salesman problem. In: 14th IEEE ISDA, pp 76–81

Hosseinabadi AR, Siar H, Shamshirband S, Shojafar M, Nizam MH, Nasir M (2015) Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in Small and Medium Enterprises. Ann Oper Res 229(1):451–474

Hosseinabadi AR, Vahidi J, Balas VE, Mirkamali SS (2016a) OVRP_GELS: solving open vehicle routing problem using the gravitational emulation local search algorithm. Neural Comput Appl 29(10):955–968

Hosseinabadi AR, Kardgar M, Shojafar M, Shamshirband Sh, Abraham A (2016b) Gravitational search algorithm to solve open vehicle routing problem. In: 6th international conference on innovations in bio-inspired computing and applications (IBICA 2015). Chapter advances in intelligent systems and computing, Kochi, India. Springer, pp 93–103

Hosseinabadi AR, Alavipour F, Shamshirbnd Sh, Balas VE (2017a) A novel meta-heuristic combinatory method for solving capacitated vehicle location-routing problem with hard time windows. Springer International Publishing Switzerland (2017) Transportation systems, advances in intelligent systems and computing, vol 454. Springer, China, pp 707–728

Hosseinabadi AR, Rostami NSH, Kardgar M, Mirkamali SS, Abraham A (2017b) A new efficient approach for solving the capacitated vehicle routing problem using the gravitational emulation local search algorithm. Appl Math Model 49:663–679

Karagöz S, Yıldız AR (2017) A comparison of recent metaheuristic algorithms for crashworthiness optimisation of vehicle thin-walled tubes considering sheet metal forming effects. Int J Veh Des 73:179–188

Khuri S, Miryala SR (1999) Genetic algorithms for solving open-shop scheduling problems. In: Proceedings of the 9th Portuguese conference on artificial intelligence. Lecture Notes in Computer Science, pp 357–368

Koulamas C, Kyparisis GJ (2015) The three-machine proportionate Open-shop and mixed shop minimum makespan problems. Eur J Oper Res 243:70–74

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1993) Sequencing and scheduling: algorithms and complexity. In: Graves SC, Rinnooy Kan AHG, Zipkin PH (eds) Handbook in operations research and management science, logistics of production and inventory, 4th edn. North-Holland, Amsterdam, pp 445–522

Low C, Yeh Y (2009) Genetic algorithm-based heuristics for an open-shop scheduling problem with setup, processing, and removal times separated. Robot Comput Integr Manuf 25:314–322

Mirmohammadi SH, Babaee Tirkolaee E, Goli A, Dehnavi-Arani S (2017) The periodic green vehicle routing problem with considering of time-dependent urban traffic and time windows. Iran Univ Sci Technol 7(1):143–156

Noori-Darvish S, Tavakkoli-Moghaddam R (2011) Solving a bi-objective open-shop scheduling problem with fuzzy parameters. J Appl Oper Res 3:59–74

Pinedo M (1995) Scheduling: theory algorithms and systems. Prentice-Hall, Englewood Cliffs

Prins C (2000) Competitive genetic algorithms for the open-shop scheduling problem. Math Methods Oper Res 52:389–411

Rostami AS, Mohanna F, Keshavarz H, Hosseinabadi AR (2015) Solving multiple traveling salesman problem using the gravitational emulation local search algorithm. Appl Math Inf Sci 9(2):699–709

Shamshirband S, Shojafar M, Hosseinabadi AR, Kardgar M, Nizam MH, Nasir M, Ahmad R (2015a) OSGA: genetic-based open-shop scheduling with consideration of machine maintenance in small and medium enterprises. Ann Oper Res 229(1):743–758

Shamshirband S, Shojafar M, Hosseinabadi AR, Abraham A (2015b) OVRP_ICA: an imperialist-based optimization algorithm for the open vehicle routing problem. In: International conference on hybrid artificial intelligence systems (HAIS), Chapter Springer LNCS, vol 9121, pp 221–233

Shojafar M, Kardgar M, Hosseinabadi AR, Shamshirband Sh, Abraham A (2016) TETS: a genetic-based scheduler in cloud computing to decrease energy and makespan. In: The 15th international conference on hybrid intelligent systems (HIS 2015), Chapter advances in intelligent systems and computing, vol 420. Seoul, South Korea. Springer, pp 103–115

Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285

Talbi, El-Ghazali (2009) Metaheuristics: from design to impelementation. Wiley, Hoboken. ISBN: 978-0-470-27858-1, 1–624

Tavakkolai H, Hosseinabadi AR, Yadollahi M, Mohammadpour T (2015) Using gravitational search algorithm for in advance reservation of resources in solving the scheduling problem of works in workflow workshop environment. Indian J Sci Technol 8(11):1–16

Tellache NEH, Boudhar M (2017) Open shop scheduling problems with conflict graphs. Discrete Appl Math 227:103–120

Tirkolaee EB, Goli A, Bakhshi M, Mahdavi I (2017) Robust multi-trip vehicle routing problem of perishable products with intermediate depots and time windows. Numer Algebra Control Optim 7(4):417–433

Tirkolaee EB, Alinaghian A, Hosseinabadi AAR, Sasi MB, Sangaiah AK (2018) An improved ant colony optimization for the multi-trip Capacitated Arc Routing Problem. Comput Electr Eng. ISSN 0045-7906. https://doi.org/10.1016/j.compeleceng.2018.01.040

Vasant P (2014) Handbook of research on artificial intelligence techniques and algorithms. IGI Global 1–796(ISBN13):9781466672581

Vasant P, Wilhelm Weber G, Dieu VN (2016) Handbook of research modern optimization algorithms and applications in engineering and economics. IGI Global 1–960(ISBN13):9781466696440

Yildiz AR (2012) A comparative study of population-based optimization algorithms for turning operations. Inf Sci 210:81–88

Yıldız AR, Ozturk N, Kaya N, Ozturk F (2007) Hybrid multi-objective shape design optimization using Taguchi's method and genetic algorithm. Struct Multidiscip Optim 34:317–332

Zhang ZH, Bai D (2014) An extended study on an open-shop scheduling problem using the minimisation of the sum of quadratic completion times. Appl Math Comput 230:238–247