Hybrid Metaheuristics for Job Shop Scheduling Problems

Cecilia E. Nugraheni, *Member IAENG*, D. Swastiani, and L. Abednego, *Member IAENG*

***Abstract*— Job Shop Scheduling (JSS) is one of the problems in the production process. The sequence of operation and processing time are often different because some jobs consist of multiple processes, with each being performed by a different machine. The purpose of JSS is to determine the order in which jobs are processed to meet specific optimization criteria. Several solutions to Job Shop Scheduling Problem (JSSP) have been proposed, either with exact approaches or heuristics. It was observed that one of the most widely used heuristic approaches is metaheuristics, specifically the Genetic Algorithm, which has the advantage of finding solutions globally, also known as global optimal. However, this algorithm is often trapped in a search that only involves local optimal values. This study proposes an approach to improve the Genetic Algorithm performance by combining it with another metaheuristic, known as the Firefly Algorithm, which has the advantage of finding solutions locally or called local optimal. It is, therefore, possible to maintain global and local optimal balance and obtain better performance by combining these two algorithms as well. Furthermore, two approaches are proposed, which include S-GAFA and C-GAFA. Several experiments were performed to measure these proposed algorithms. It was observed that S-GAFA and C-GAFA performed better than the Genetic Algorithm in solving JSSP.**

***Index Terms*— Firefly Algorithm, Genetic Algorithm, Job Shop Scheduling Problem, metaheuristic, optimization**

I. Introduction

There are several types of scheduling problems in the industry based on job processing flow. One of them is Job Shop Scheduling Problem (JSSP), in which each job moves from one machine to another in a non-homogeneous pattern.

JSSP is a combinatorial optimization problem, which is a topic in theoretical computer science and applied mathematics. Furthermore, it consists of finding the most cost-effective solutions to mathematical problems. Many solutions to JSSP have been proposed, either with an exact algorithm or heuristics. It has been observed that one of the most widely used heuristic approaches to this problem is metaheuristics which are a class of intelligent self-learning algorithms for finding near-optimal solutions to hard optimization problems, mimicking intelligent processes, and behaviors observed in nature, sociology, reasoning, and other disciplines [1]. Several metaheuristics classes have also been proposed, which include evolutionary algorithms [2]-[10], swarm intelligence-based approaches [11]-[17], and local search algorithms [18]-[20].

The two examples of algorithms that are included in metaheuristics are the Genetic Algorithm (GA) and the Firefly Algorithm (FA). The GA is a computational algorithm inspired by Darwin's theory of evolution, and it states that organisms with high fitness values tend to survive, while those with low values die. This theory was incorporated into a computational algorithm to solve problems in a more "natural" way. Ever since John Holland first pioneered this theory, GA has been studied and widely applied in various fields. For example, it has been used in practical problems that focus on determining optimal parameters.

Furthermore, the FA is an optimization algorithm inspired by the blinking behavior of fireflies. The main purpose of a firefly's blinking behavior is to be a signal for attracting other fireflies. This means that a firefly with a brighter light tends to attract others that have a lower light.

The advantage of GA is that it can find solutions globally, known as global optimal. However, this algorithm is often trapped in a search that only involves local optimal values because it starts optimization from an initial point and only aims for maximum values in a particular area. This is unlike the FA, as it finds solutions locally or known as local optimal. This means that the two algorithms can maintain the balance between global and local optimal. Based on these aforementioned advantages, several studies tried to combine the two algorithms. This combination aims to obtain a new approach that has better performance than either of these algorithms working individually.

The FA has been used in several methods and approaches to solve many variants of JSSP [21]-[28]. In [29], Wahid et.al. proposed a combination of the GA and FA by introducing GA operators, which include selection, mutation, and crossover operators at the positioning stage of the standard FA. This combination aims to overcome the FA's weaknesses, such as unbalanced exploration and exploitation. The proposed algorithm was further applied to optimize energy consumption and user comfort management in smart buildings. In [30], a novel method for the traveling salesman problem was proposed based on a discrete FA with a combination of GA. The distance of the FA was redefined by introducing swap operators and sequences to prevent the algorithm from falling easily into the local solutions and to

Manuscript received September 22, 2021; revised October 12, 2022. This work was supported in part by the Indonesian Ministry of Research, Technology and Higher Education (RistekDikti), under the research scheme of Higher Education Excellence Applied Research 2019-2021 with contract number: III/LPPM/2020-04/104-PE-S.

Cecilia E. Nugraheni is a lecturer at the Informatics Dept., Fac. of Information Technology and Science, Parahyangan Catholic University, Bandung, Indonesia (email:cheni@unpar.ac.id).

Dian Swastiani is a postgraduate student majoring in Informatics at Bandung Institute of Technology, Indonesia (email: dian.swastiani@gmail.com).

Luciana Abednego is a lecturer at the Informatics Dept., Fac. of Information Technology and Science, Parahyangan Catholic University, Bandung, Indonesia (email:luciana@unpar.ac.id).

accelerate convergence speed. It was also observed that several other studies used a combination of metaheuristics to solve JSSP [31]-[49].

In this study, the use of the FA was investigated to improve the performance of the GA in solving JSSP. Two algorithms were proposed to combine the Genetic and the Firefly, and the optimization criteria were the makespan. Even though there are many similar works, this current study differs in terms of the application order of GA and FA, chromosome modeling, methods for mutation and crossover operations, and the principle of firefly movement.

II. Methodology

A. Job Shop Scheduling Problems (JSSP)

Given n jobs, $J = \{J_1, \ldots, J_n\}$ with different processing times, which are to be assigned to m machines $M = \{M_1, ..., M_m\}$. The process order for each job is often different, and every job is processed exactly once on every machine. Furthermore, each job (J_i) consists of m operations $\{O_{i1}, \ldots, O_{im}\}$, and every operation has a different processing time PT_{ij}, for a job i at machine j. The goal is to find a schedule with minimum makespan. It is important to note that makespan is the total time needed until all these jobs are ready to be processed on these machines.

Table I illustrates a small JSSP consisting of three jobs and three machines. The first job, J_1, consists of three operations which include O_{11}, O_{12}, and O_{13}, which are processed on machines M_1, M_2, and M_3 for 10, 9, and 8 units of time, respectively. The second and third jobs, J_2 and J_3, also consist of three operations, but the machines' order for the third job was different. For example, M_3 was for the first operation, M_1 was for the second, and M_2 was for the third.

TABLE I
AN EXAMPLE OF JOB SHOP SCHEDULING PROBLEMS

Job	Operation i1		Operation i2		Operation i3	
	Time	*Machine*	*Time*	*Machine*	*Time*	*Machine*
J_1	10	1	9	2	8	3
J_2	9	3	8	1	7	2
J_3	10	3	8	1	11	2

Fig. 1 shows one of the possible schedules for this problem, in which the order of jobs processed by M_1 is J_1 (O_{11}), J_2 (O_{21}), and J_3 (O_{31}). The second machine, M_2, also has the same sequence, namely J_1 (O_{12}), J_2 (O_{22}), and J_3 (O_{32}). Meanwhile, M_3 works on J_2 (O_{23}), followed by J_3 (O_{33}), and J_1 (O_{13}). The makespan generated by this schedule is 38 units of time.

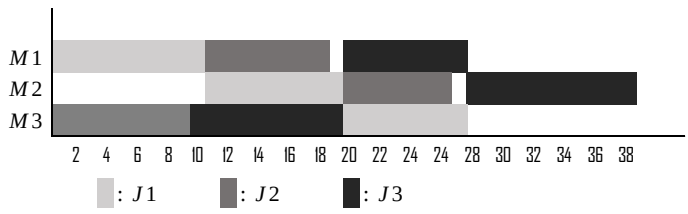

Fig. 1. The Gantt Chart for a solution of JSSP in Table I.

B. Genetic Algorithms (GA)

GA is often used to address optimization problems, but it can also solve other types of problems. John Holland stated that it is possible to formulate every problem in adaptation such as natural or artificial, using genetic technology. This is because GA simulates the Darwinian evolutionary process and genetic operations on chromosomes.

Furthermore, the solution to the problem is represented in GA as a chromosome, which consists of multiple genes. Each gene contains a value called an allele. It is important to note that the modeling of chromosomes is specific to the problem in this study.

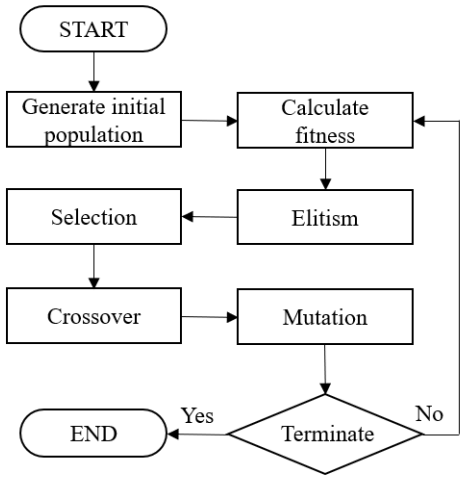

Fig. 2. The Genetic Algorithm's Flowchart.

Fig. 2 shows the steps taken in GA. The first step is to generate the initial population by creating a set of chromosomes, which are the first candidate solutions. The algorithm repeatedly defines the new populations, starting with the initial population until the termination conditions are met. At the beginning of each iteration, the quality of each chromosome was assessed using a function called fitness. Optionally, an elitism process was performed to preserve the best candidate solutions in the new population by selecting some of the best chromosomes, which are further selected using a selection mechanism.

Furthermore, two genetic operations are conducted based on several established criteria, which include crossover and mutation rates. The obtained chromosomes from these operations were inserted into the new population. The termination was then performed depending on the specified number of generations or particular conditions.

C. Firefly Algorithm (FA)

FA is a metaheuristic algorithm inspired by the flickering behavior of fireflies. There are two main functions of flashing lights, which include attracting the attention of other fireflies and attracting prey. Xin-She Yang developed this algorithm in 2007, by using the following three rules [24]:

1. All fireflies are the same gender, meaning that one firefly tends to be attracted to others regardless of gender.
2. The attraction is proportional to the brightness of the

firefly's light. Therefore, fireflies with lower brightness levels tend to be attracted and move to those with higher brightness. It is important to note that brightness tends to decrease with distance and light absorption due to the air factor. When none of the fireflies has the brightest light among the population, they all move randomly.

3. The brightness of a firefly is influenced by the value of the objective function of the given problem. In the maximization problem, the brightness is proportional to the value of the objective function.

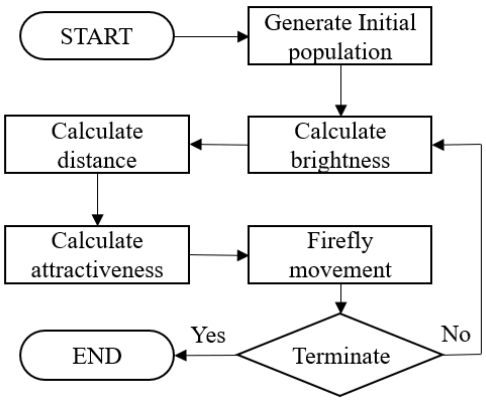

Fig. 3. The Firefly Algorithm's Flowchart.

According to Fig. 3, the first step in FA is to initialize a set of candidate solutions, which is a firefly. The next step is to calculate the brightness of each firefly that reflects the corresponding quality. The following steps are three calculations, 1) the distance between a pair of fireflies, 2) the attractiveness based on the brightness and the distance between the fireflies, and 3) the movement calculation of the fireflies.

D. The Proposed Algorithms

In this work, two ways for combining GA with FA were proposed. The first way was to run GA and FA sequentially, in which the final population generated by GA becomes the initial population of FA. It was observed that the final population of GA contained the best solution generated by the algorithm. This condition is achieved by using the principle of elitism. Furthermore, the algorithm is named S-GAFA, which stands for Sequential Hybrid Genetic Algorithm and Firefly Algorithm. The working of this algorithm is given in the flowchart shown in Fig. 4.

The second way was to run both algorithms in each search iteration for a new set of solutions. After generating the initial population, steps are formulated from GA to produce a new one, which was further subjected to processes derived from FA. The population generated by FA becomes the input for GA and vice versa, in order to obtain a new result. These stages are performed iteratively until a termination condition was met. This algorithm is called C-GAFA, which stands for Cyclic Hybrid Genetic Algorithm and Firefly Algorithm. The working of this algorithm is given in the flowchart shown in Fig. 5.

Chromosomes and fireflies are represented as arrays, with each element being a triple $\langle x,y,z\rangle$ where x, y, z are integers representing job, operation, and machine. For example, $\langle 1,3,2\rangle$ represents the third operation of job 1 processed by machine 2. The length of the array is the number of operations.

The following equation was used to model the solution quality, fitness in GA, and brightness in FA

$$f(x) = \frac{1}{Cmax_x + 1} \tag{1}$$

where x is a chromosome/firefly and $Cmax_x$ is its corresponding makespan.

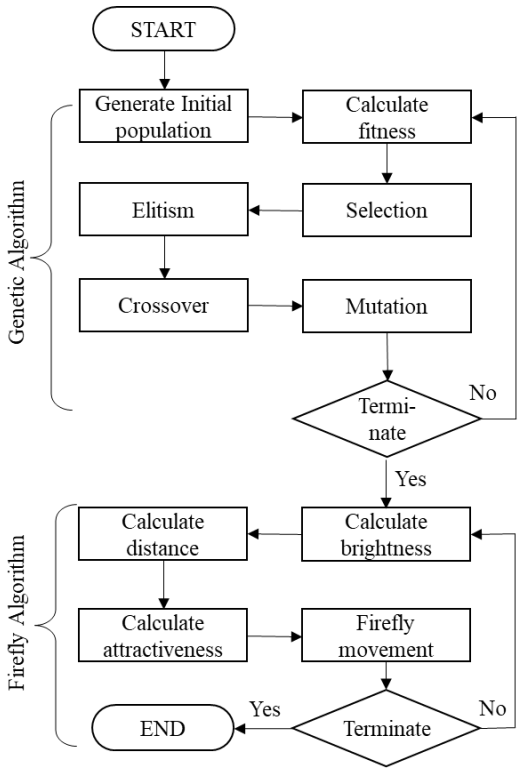

Fig. 4. The S-GAFA Algorithm's Flowchart.

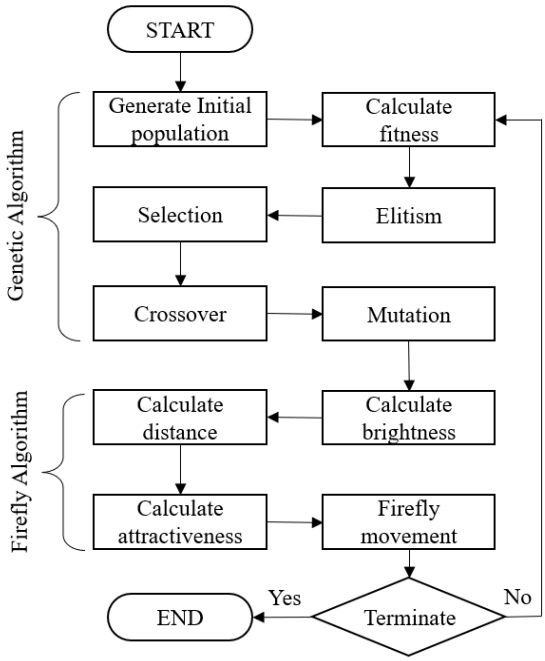

Fig. 5. The C-GAFA Algorithm's Flowchart.

Furthermore, the mechanism of elitism takes the best chromosomes from the current population and inserts them directly into the new population. These chromosomes are not involved in the process of crossover or mutation. In the selection process, the roulette wheel technique was employed.

A random number was generated in each iteration to determine whether a crossover was performed. The two best chromosomes are subjected to crossover operations when the random number was smaller than the predetermined *crossover rate* (CR). In [24], the technique used was a modified one-point crossover, and the process was explained in Fig. 5. After the cross point was determined i.e. the bold vertical line, the genes on the right side of the line were only exchanged with their machine numbers. It was this exchange of partial gene values that distinguished it from the usual crossover methods.

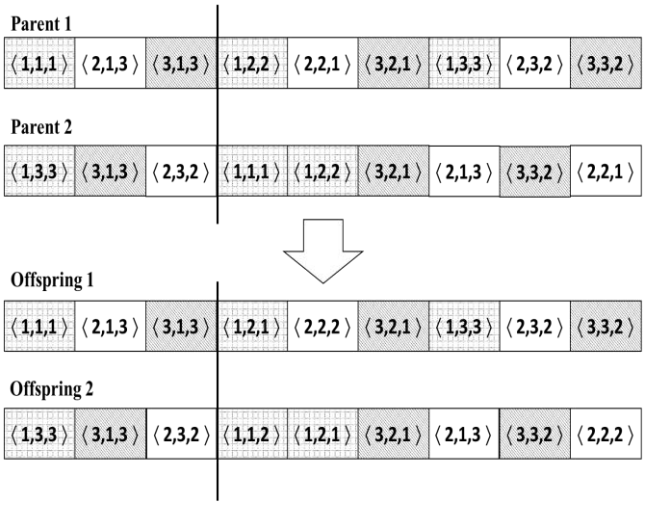

Fig. 5. A modified one-point crossover.

Furthermore, a random number was generated in the crossover operation to determine whether a mutation was performed. A randomly selected chromosome undergoes a mutation operation when the random number is smaller than the predetermined *mutation rate* (MR). Two genes from the selected chromosome are randomly selected and exchanged. For example, the third and the seventh genes are exchanged, thereby resulting in a new chromosome as shown in Fig. 6.

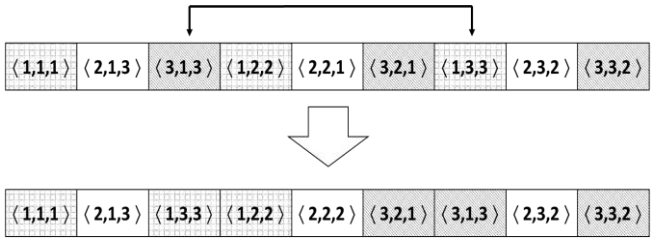

Fig. 6. A mutation.

The distance between two fireflies is the number of elements exchanged in the first firefly array for it to have the same order of elements as the second firefly.

The attractiveness of a firefly f_1 over firefly f_2 is calculated by using the equation (2):

$$\beta = \frac{\beta_0}{1+\gamma r^2} \tag{2}$$

Where β is the attractiveness, r is the distance between f_1 and f_2, β_0 is the firefly attraction value (base beta), and γ is the coefficient of light absorption, for $0 \leq \beta_0 \leq 1$ and $0 \leq \gamma \leq 1$.

The last process in FA is firefly movements and based on [25], the two-step movement principle was used as follows. The brightness of all pairs of fireflies in the population was compared, and a random number was generated for those with a lower brightness level. When this random number is smaller than the brightness value, the firefly moves or changes position. The calculation of the distance between two fireflies shows the elements that need to be swapped for the elements' order of the array to be the same. Consequently, a random number was generated for each pair of elements. It was observed that when the value of this random number is smaller than the brightness value, the element is swapped. The second step movement is the exchange of elements such as the GA mutation process. Final tests are performed by checking the number of iterations or generations set in the initial stage.

III. Experimental Results

The experiments performed in this study are described as follows. The purpose is to reveal whether it is possible to improve the performance of GA by combining it with FA. Specifically, it aims to find out whether S-GAFA and C-GAFA have better performance than GA. The influence of GA parameters on its performance was also analyzed. In this study, the parameter tested is the crossover rate, but the mutation was not tested based on the assumption that its effect was much smaller than the crossover.

A. Experiment Setting

The sample problems are obtained from the benchmarks commonly used to test JSSP [50][51]. There were 55 problems in total, of which 5 examples are from Adam et al., while 10 and 40 are found in Applegate and Cook, and Lawrence, respectively. Each instance has a different size i.e. number of jobs and number of machines, as shown in Table II.

Table III shows the parameter values used in the experiment. With reference to [50], the FA parameter suggested for gamma and beta values are 1 and 0.1, respectively. *Run* represents the number of makespan calculations performed by each algorithm or crossover value for every problem instance.

Furthermore, four algorithms and three crossover rate values were used to compute the makespan of each instance for 100 times. A total of 1000 computation results were obtained for each instance because ten algorithm variations were executed, namely FA, GA-0.70, GA-0.75, GA-0.80, S-GAFA-0.70, S-GAFA-0.75, C-GAFA-0.80, C-GAFA-0.70, C-GAFA-0.75, and C-GAFA-0.80. Out of the resulting 1000 makespans for each problem instance, the average, minimum, and maximum values were calculated.

TABLE II
PROBLEM INSTANCES

Benchmark	Instances	Problem Size (job x machine)
Adam et.al. (1988)	abz5 - abz6	10 x 10
	abz7 - abz9	20 x 15
Applegate & Cook (1991)	orb01 - orb10	10 x 10
Lawrence (1984)	la01 – la05	10 x 5
	la06 – la10	15 x 5
	la11 – la15	20 x 5
	la16 – la20	10 x 10
	la21 – la25	15 x 10
	la26 – la30	20 x 10
	la31 – la35	30 x 10
	la36 – la40	15 x 15

TABLE III
PARAMETER SETTING

No	Parameter	Value
1	Crossover Rate	{0.70, 0.75, 0.80}
2	Mutation Rate	0.01
3	Gamma	1.0
4	Beta	0.1
5	Population Size	50
6	Generation	50
7	Run	100

Furthermore, four algorithms and three crossover rate values were used to compute the makespan of each instance for 100 times. A total of 1000 computation results were obtained for each instance because ten algorithm variations were executed, namely FA, GA-0.70, GA-0.75, GA-0.80, S-GAFA-0.70, S-GAFA-0.75, C-GAFA-0.80, C-GAFA-0.70, C-GAFA-0.75, and C-GAFA-0.80. Out of the resulting 1000 makespans for each problem instance, the average, minimum, and maximum values were calculated.

B. Results and Analysis

Table IV shows the average makespan generated by each algorithm. Meanwhile, Table V shows the minimum and maximum makespan values produced by each algorithm for every problem instance.

TABLE IV
AVERAGE MAKESPAN RESULTED BY EACH ALGORITHM

Instance	Benchmark	FA	GA	S-GAFA	C-GAFA
abz5	1234	1598	1717	1601	1603
abz6	943	1223	1320	1218	1222
abz7	667	1073	1133	1071	1071
abz8	670	1104	1167	1104	1105
abz9	691	1120	1181	1119	1119
orb01	1059	1504	1614	1506	1512
orb02	888	1216	1310	1218	1216
orb03	1005	1519	1628	1518	1522
orb04	1005	1361	1453	1361	1360
orb05	887	1286	1385	1286	1287
orb06	1010	1470	1579	1468	1475
orb07	397	550	593	550	550
orb08	899	1336	1435	1333	1336
orb09	934	1273	1367	1271	1272
orb10	944	1383	1492	1378	1384
la01	666	767	822	764	772
la02	655	807	863	809	815
la03	597	736	785	735	739
la04	590	732	790	733	734
la05	593	613	660	614	617
la06	926	1018	1092	1016	1025
la07	890	1068	1140	1068	1078
la08	863	1021	1094	1017	1027
la09	951	1052	1134	1055	1065
la10	958	1012	1085	1014	1017
la11	1222	1364	1450	1364	1375
la12	1039	1175	1256	1175	1181
la13	1150	1298	1387	1298	1304
la14	1292	1345	1426	1340	1353
la15	1207	1503	1584	1498	1514
la16	945	1205	1290	1201	1213
la17	784	1023	1108	1021	1028
la18	848	1107	1187	1103	1111
la19	842	1134	1227	1134	1145
la20	902	1180	1275	1180	1189
la21	1046	1542	1652	1544	1555
la22	927	1389	1496	1390	1405
la23	1032	1477	1597	1482	1488
la24	935	1396	1505	1391	1405
la25	977	1433	1532	1431	1440
la26	1218	1840	1968	1840	1854
la27	1235	1898	2020	1899	1909
la28	1216	1834	1965	1838	1855
la29	1153	1828	1945	1822	1825
la30	1355	1952	2068	1943	1947
la31	1784	2491	2623	2489	2486
la32	1850	2650	2792	2648	2648
la33	1719	2420	2554	2416	2421
la34	1721	2481	2615	2478	2474
la35	1888	2679	2823	2680	2680
la36	1268	1905	2032	1907	1906
la37	1397	2121	2245	2115	2118
la38	1196	1876	1997	1877	1876
la39	1233	1923	2051	1924	1928
la40	1222	1907	2032	1904	1908

TABLE V
MINIMUM AND MAXIMUM MAKESPANS RESULTED BY EACH ALGORITHM

	FA		GA		S-GAFA		C-GAFA	
Inst.	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
abz5	1459	1704	1559	1873	1458	1715	1445	1721
abz6	1143	1315	1171	1450	1111	1313	1094	1314
abz7	997	1130	1046	1213	980	1141	990	1134
abz8	1021	1165	1059	1238	1018	1166	1027	1166
abz9	1032	1171	1046	1266	1024	1185	1045	1172
orb01	1394	1607	1443	1758	1391	1589	1393	1589
orb02	1094	1294	1146	1436	1118	1298	1096	1300
orb03	1347	1612	1374	1798	1374	1623	1371	1624
orb04	1262	1458	1311	1589	1249	1454	1262	1442
orb05	1159	1383	1246	1508	1171	1377	1133	1403
orb06	1341	1578	1321	1731	1311	1578	1326	1572
orb07	488	587	502	651	502	589	495	588
orb08	1199	1425	1267	1573	1174	1424	1219	1420
orb09	1162	1357	1206	1493	1146	1353	1115	1379
orb10	1223	1477	1275	1648	1203	1474	1213	1498
la01	699	837	708	910	693	819	722	823
la02	747	861	771	947	751	872	745	859
la03	680	782	696	858	669	789	669	792
la04	653	784	687	893	667	798	652	806
la05	593	665	593	749	593	665	593	663
la06	926	1084	997	1199	939	1103	956	1090
la07	992	1144	1003	1238	970	1146	984	1155
la08	943	1111	966	1219	944	1086	927	1090
la09	985	1118	1010	1242	951	1134	972	1134
la10	958	1090	985	1209	958	1090	958	1069
la11	1270	1460	1299	1575	1246	1443	1272	1449
la12	1099	1246	1115	1387	1065	1256	1061	1246
la13	1178	1392	1233	1534	1172	1387	1186	1372
la14	1292	1412	1292	1546	1292	1417	1292	1450
la15	1378	1582	1441	1717	1386	1581	1439	1588
la16	1115	1302	1142	1398	1094	1283	1104	1280
la17	918	1100	970	1209	894	1109	930	1123
la18	995	1193	1063	1338	986	1186	1018	1195
la19	1034	1215	1108	1332	1047	1215	1035	1242
la20	1066	1270	1127	1388	1079	1291	1067	1278
la21	1402	1656	1496	1796	1391	1657	1385	1643
la22	1292	1475	1320	1635	1278	1486	1253	1482
la23	1360	1579	1418	1762	1340	1585	1345	1578
la24	1275	1503	1332	1636	1264	1522	1262	1497
la25	1290	1540	1352	1695	1285	1528	1327	1534
la26	1717	1984	1800	2110	1696	1965	1661	1970
la27	1727	2020	1796	2188	1743	2016	1735	2014
la28	1682	1942	1811	2117	1666	1957	1691	1952
la29	1676	1943	1744	2086	1680	1942	1628	1935
la30	1810	2077	1819	2235	1805	2086	1813	2049
la31	2261	2634	2285	2797	2285	2616	2314	2606
la32	2478	2787	2530	2944	2424	2802	2449	2795
la33	2185	2574	2301	2739	2182	2535	2171	2561
la34	2307	2608	2400	2787	2222	2647	2317	2626
la35	2438	2826	2546	3042	2481	2826	2503	2910
la36	1756	2017	1869	2179	1721	2019	1739	2019
la37	1900	2232	1984	2412	1932	2231	1965	2262
la38	1696	1991	1801	2151	1649	2011	1661	1974
la39	1754	2042	1809	2200	1767	2041	1773	2039
la40	1710	2029	1738	2192	1699	2002	1762	2021

Furthermore, Table IV contains the best values that have been previously reported. These values are used as a reference to calculate the *Relative Error* (RE), which is the difference between the value generated by each algorithm and the value from the benchmark. For each instance *i* and each algorithm *A*, the RE of *i* is calculated by using this formula:

$$RE_i = \frac{A_i - B_i}{B_i} * 100\% \tag{3}$$

where:

- RE_i denotes the relative error of makespan of instance *i* produced by algorithm *A*,
- A_i denotes the makespan of instance *i* produced by algorithm A,
- B_i denotes the benchmark's value for instance *i*.

The Mean Relative Error (*MRE*) was calculated from several REs using the following formula:

$$MRE = \frac{1}{n}\sum_{i=1}^{n} RE_i * 100\% \tag{4}$$

where *n* is the number of REs.

Fig. 7 shows the MRE comparison of each algorithm, which are divided into three groups, namely the average, minimum, and maximum, and they showed the same tendency in term of the order. For example, the algorithms are arranged in descending order based on MRE, such as GA, C-GAFA, S-GAFA, and FA. This means that GA has the worst performance and FA is the best, meanwhile, C-GAFA and S-GAFA have better performance than GA. It was also observed that the combination of GA and FA improved the performance of GA, while S-GAFA performed better than C-GAFA.

Fig. 8 shows the comparison of MRE for each crossover rate value for GA, S-GAFA, and C-GAFA algorithms. It was observed that the crossover rate value of 0.70 has the worst results, while the other two values, namely 0.75 and 0.80, do not form a regular pattern. Furthermore, the 0.75 crossover rate in GA produced better results than 0.80, but the situation recorded for the other two algorithms was the opposite.

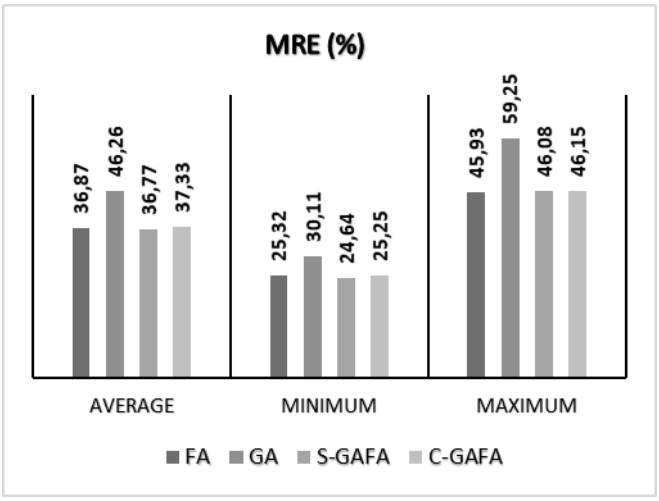

Fig. 7. MRE of each algorithm.

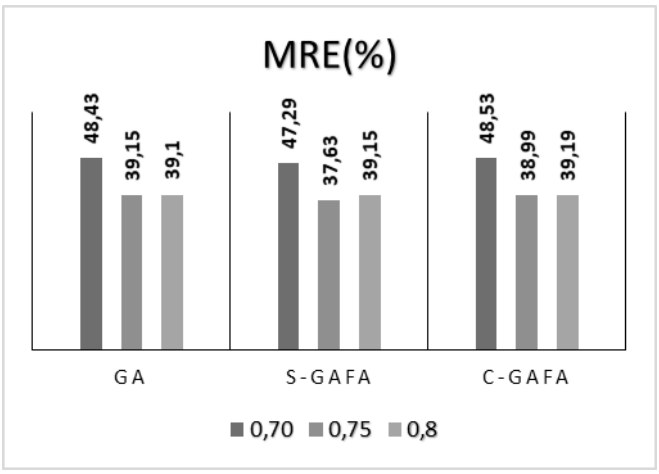

Fig. 8. MRE of each crossover rate.

IV. Conclusion

In this study, the performance improvement of GA was investigated using FA, and two algorithms were proposed, which include S-GAFA and C-GAFA. In the S-GAFA, the two algorithms are run sequentially, starting with GA, and further continuing with FA. Meanwhile, GA and FA are combined in C-GAFA by executing GA, and iteratively running FA.

These algorithms were utilized to solve JSSP. Accordingly, several adjustments have been made, such as chromosomes and fireflies modeling, methods or techniques for crossover, mutation, and movement of fireflies.

The results show that S-GAFA and C-GAFA performed better than GA, and it supports the hypothesis. Furthermore, S-GAFA gave better results than C-GAFA, but the crossover rate value of 0.70 resulted in the worst performance among the three values tested.

This study only focused on improving the performance of GA by using FA, but the makespan calculation results or the quality of the resulting solution have not been considered. It was observed that the solution qualities are still inferior to the reference value, even though the two proposed algorithms have succeeded in improving the performance of GA. Therefore, further studies are needed to determine the appropriate parameter values of each basic algorithm, such as the crossover rate, mutation rate, gamma, and beta values.

References

[1] K-L. Du and M.N.S. Swamy, “Search and optimization by metaheuristics techniques and algorithms inspired by nature”, Birkhäuser. 2016.

[2] F. A. Toader, "Evolutionary algorithms for job shop scheduling", Proceedings of the Conference on Electronics, Computers and Artificial Intelligence, pp 1-6, 2016.

[3] J. Ding, Z. Lü, C.-M. Li, L. Shen, L. Xu, and F. Glover, “A two-individual based evolutionary algorithm for the flexible job shop scheduling problem”, AAAI, vol. 33, no. 01, pp 2262-2271, 2019.

[4] P. Sriboonchandr, N. Kriengkorakot, P. Kriengkorakot, “Improved differential evolution algorithm for flexible job shop scheduling problems”, Math. Comput. Appl., vol. 24, no. 80, 2019.

[5] Y. Wang and Q. Zhu, "A hybrid genetic algorithm for flexible job shop scheduling problem with sequence-dependent setup times and job lag times," in IEEE Access, vol. 9, pp 104864-104873, 2021.

[6] I. A. Chaudhry, "A genetic algorithm approach for process planning and scheduling in job shop environment", Proceedings of the World Congress on Engineering 2012, Vol. III WCE 2012, 2012, London, U.K.

[7] H. M. Abd-Elaziz, M.A. Awad, F. Tolba, ”Integrated process planning and scheduling in smart manufacturing using genetic-based algorithm”, International Journal of Scientific & Technology Research vol. 10, issue 03, 2021.

[8] M. Gabli, E. M. Jaara, and E. B. Mermri, “A genetic algorithm approach for an equitable treatment of objective functions in multi-objective optimization problems”, IAENG International Journal of Computer Science, vol. 41, no. 2, pp 102-111, 2014.

[9] S. Zhang, Z. Yu, W. Zhang, D. Yu, D. Zhang, "Distributed integration of process planning and scheduling using an enhanced genetic algorithm”, International Journal of Computing, Information, and Control, vol. 11, no. 5, 2015.

[10] E. Haq, I. Ahmad, A. Hussain, I. M. Almanjahie, "A novel selection approach for genetic algorithms for global optimization of multimodal continuous functions", Journal of Computational Intelligence and Neuroscience, vol. 2019, pp 1-14, 2019.

[11] S. Kavitha, P. Venkumar, “Flexible job shop scheduling using hybrid swarm intelligence”, International Journal of Engineering and Advanced Technology, vol. 9, issue 154, pp 2249 – 8958, 2019.

[12] Z. Wang, J. Zhang, S. Yang, “An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals”, Swarm and Evolutionary Computation, vol. 51, no. 5, 2019.

[13] P. Fattahi, N. B. Rad, F. Daneshamooz, and S. Ahmadi, "A new hybrid particle swarm optimization and parallel variable neighborhood search algorithm for flexible job-shop scheduling with assembly process", Assembly Automation, vol. 40, no. 3, pp 419-432, 2020.

[14] J. M. Todd, K. Steinhöfel, and P. Veenstra, “Firefly-inspired algorithm for job shop Scheduling” in H.-J. Böckenhauer et al. (Eds.): Hromkovíc Festschrift, LNCS 11011, pp. 423–433, 2018.

[15] Y. Halim, C.E. Nugraheni, “A bee colony algorithm based solver for flow shop scheduling problem”. Int. J. Inform. Visualization, vol. 5, no. 2, pp 170-176, 2021

[16] L. Asadzadeh, “Solving the job shop scheduling problem with an enhanced artificial bee colony algorithm through local search heuristic”, Recent Advances in Computer Science and Communications, vol. 14, no. 7, 2021.

[17] D. Y. Sha, Hsing-Hung Lin, C.-Y. Hsu, "A modified particle swarm optimization for multi-objective open shop scheduling", Proceedings of the International Multiconference of Engineers and Computer Scientist 2010 Vol III, pp 17-19, 2010.

[18] J.C. Seck-Tuoh-Mora, N.J. Escamilla-Serna, J. Medina-Marin, N. Hernandez-Romero, I. Barragan-Vite, J.R. Corona-Armenta, “A global-local neighborhood search algorithm and tabu search for flexible job shop scheduling problem”, PeerJ Comput Sci., vol. 7, no. 574, 2021

[19] S. Kosasih, C.E. Nugraheni, L. Abednego, "Artificial immune system applied to job shop scheduling", Journal of Industrial and Intelligent Information, vol. 9, no. 1, pp 15-22, 2021.

[20] X. Wang, L. Gao, C. Zhang, and X. Shao. 2010. “A multiobjective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problems”, The International Journal of Advanced Manufacturing Technology, vol. 51, no. 5, pp. 757–767, 2010.

[21] K.C. Udaiyakumara, M. Chandrasekaran, “Application of firefly algorithm in job shop scheduling problem for minimization of makespan”, Procedia Engineering, vol. 97, pp 1798 – 1807, 2014.
[22] R. Luo, L. Liu, D. Tan, and S. Yin, "Scheduling feature selection for data-driven job shop scheduling system using improved firefly algorithm optimization," in 2019 International Conference on High-Performance Big Data and Intelligent, pp 116-121, 2019.
[23] D.B. Sari, S. Batubara, R.R. Sindyastuti, “Scheduling design for job shop production using firefly algorithm to minimize mean tardiness”, in Proc. Of Int. Conf. on Industrial Engineering and Operations Management, pp 2274- 2283, 2020.
[24] N. Á. Gil, R. Rosillo, D. de la Fuente, et al. “A discrete firefly algorithm for solving the flexible job-shop scheduling problem in a make-to-order manufacturing system”. Cent Eur J Oper Res, vol. 29, pp 1353–1374 (2021).
[25] M. Liebenlito, N. Inayah, A. N. Rahmah, and A. Widiatmoko, “Modified firefly algorithm using smallest position value for job-shop scheduling problems”, in Proceedings of the International Conference on Mathematics and Islam, pages 23-27, 2018.
[26] K. G. Devi, R. S. Mishra, A. K. Madan, “A dynamic adaptive firefly algorithm for flexible job shop scheduling”, Intelligent Automation & Soft Computing, vol. 31, no. 1, pp 429–448, 2022.
[27] H. Lo, S. Fong, Y. Zhuang, X. Wang, and T. Hanne, "Applying a chaos-based firefly algorithm to the permutation flow shop scheduling problem," in Proceeding 3rd International Symposium on Computational and Business Intelligence, pp. 51-57, 2015.
[28] G. K. Kantak, A. Singh, A. Ansari, A. Kumar, M. Singh. “Application and significance of firefly algorithm for multi-objective job shop scheduling”, in Proceedings of National Conference on Multidisciplinary Engineering Sciences and Information Technology, pp 01-03, 2020.
[29] F. Wahid, R. Ghazali, L.H. Ismail, “Improved firefly algorithm based on genetic algorithm operators for energy efficiency in smart buildings”, Arabian Journal for Science and Engineering, vol. 44, no. 4, pp 4027-4047, 2020.
[30] L. Teng, H. Li, “Modified discrete firefly algorithm combining genetic algorithm for traveling salesman problem”, TELKOMNIKA, vol 16, no 1, pp 424-431, 2018.
[31] T. C. E. Cheng & B. Peng & Z. Lü, "A hybrid evolutionary algorithm to solve the job shop scheduling problem", Annals of Operations Research, Springer, vol. 242, no. 2, pp 223-237, 2016.
[32] S. Kavitha, P. Venkumar, “Flexible job shop scheduling using hybrid swarm intelligence”, International Journal of Engineering and Advanced Technology, vol. 9, issue154, 2019.
[33] K. Benhamza, O. Zedadra, “Hybrid metaheuristic for optimization job-shop scheduling problem”, International Journal of Informatics and Applied Mathematics, vol. 1, no. 1, pp 1-9, 2018.
[34] P. Pongchairerks, "A two-level metaheuristic algorithm for the job-shop scheduling problem", Complexity, vol. 2019, pp 1-11, 2019.
[35] A. S. Eesa, A. M. Abdulazeez, Z. Orman. “A novel bio-inspired heuristic”, International Journal of Scientific and Engineering Research, vol. 4, no. 9, pp 1978-1986, 2013.
[36] A. Phu-ang, “A hybrid firefly algorithm with fuzzy movement for solving the flexible job shop scheduling problem”, Ecti Transactions on Computer and Information Technology, vol. 15, no. 2, 2021.
[37] S. Karthikeyan, P. Asokan, M. Chandrasekaran. “A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problems with maintenance activity”, in Applied Mechanics and Materials, vol. 575, pp. 922–925, 2014.
[38] M. G. Kharat, S. S. Khadke, R. Raut, S. Kamble, S.J. Kamble, & M. G. Kharat, “Application of hybrid firefly algorithm-tabu search technique to minimize the makespan in job shop scheduling problem”, International Journal of Applied Industrial Engineering, vol. 3, no. 2, pp 1–21, 2016.
[39] M. Rohaninejad, A. S. Kheirkhah, B. V. Nouri & P. Fattahi. “Two-hybrid tabu search–firefly algorithms for the capacitated job shop scheduling problem with sequence-dependent setup cost”, International Journal of Computer Integrated Manufacturing, vol. 28, no. 5, pp 470-487, 2015.
[40] A. Phu-ang, “The hybrid firefly algorithm with the fuzzy movement method for solving a complex scheduling problem”, ECTI-CIT, vol. 15, no. 2, pp 208 - 219, 2021.
[41] X. Huang, L. Yang, "A hybrid genetic algorithm for multi-objective flexible job shop scheduling problem considering transportation time", International Journal of Intelligent Computing and Cybernetics, vol. 12, no. 2, pp 154-174, 2019.
[42] M. E. Meziane, T. Noria, “A hybrid genetic algorithm with a neighborhood function for flexible job shop scheduling”, Multiagent and Grid Systems, vol. 14, no. 2, pp 161-175, 2018
[43] R. M. Branco, A. S. Coelho, S. F. Mayerle, “Hybrid genetic algorithms: solutions in realistic dynamic and setup dependent job-shop scheduling problems”, International Journal of Production Management and Engineering, vol. 4, no. 2, pp 75-85, 2016.
[44] C. Wang, W. Song, L. Liu, “An adaptive bat algorithm with memory for global optimization”, IAENG International Journal of Computer Science, vol. 45, no. 2, pp 320-327, 2018.
[45] L. Dang, Y. Hou, Q. Liu, Y. Kong, "A hybrid metaheuristic algorithm for the bi-objective school bus routing problem," IAENG International Journal of Computer Science, vol. 46, no.3, pp 409-416, 2019.
[46] S. Mousavipour, H. Farughi, F. Ahmadizar, "A job shop scheduling problem with sequence-dependent setup times considering position-based learning effects and availability constraints", International Journal of Industrial Engineering & Production Research, vol. 30, no. 3, pp 329-340, 2019
[47] S. Noor, M. I. Lali, M.S. Nawaz, “Solving job shop scheduling problem with genetic algorithm", Sci.Int.(Lahore), vol. 27, no. 4, pp 3367-3371, 2015.
[48] W. T. Lunardi, H. Voos, “Comparative study of genetic and discrete firefly algorithm for combinatorial optimization”, 33rd ACM/SIGAPP Symposium on Applied Computing, pp 1–9, 2018.
[49] J.J. van Hoorn, “The current state of bounds on benchmark instances of the job-shop scheduling problem”, J Sched, vol. 21, pp 127–128, 2018.
[50] A. Khadwilard, S. Chansombat, T. Thepphakorn, P. Thapatsuwan, W. Chainate and P. Pongcharoen, “Application of firefly algorithm and its parameter setting for job shop scheduling”, The Journal of Industrial Technology, vol. 8, no. 1, 2012.
[51] M. Chandrasekaran, P. Asokan, S. Kumanan, T. Balamurugan, “Sheep-flocks heredity model algorithm for solving job shop scheduling problems”, the International Journal of Applied Management and Technology, vol. 4, no. 1, pp 79-100, 2006.