2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

A Novel Collective Crossover Operator for Genetic Algorithms

Berna Kiraz
Department of Computer Engineering
Fatih Sultan Mehmet Vakif University
Istanbul, Turkey
bkiraz@fsm.edu.tr

Azam Asilian Bidgoli
Department of Electrical and Computer Engineering
University of Kashan
Kashan, Iran
Azam.AsilianBidgoli@ontariotechu.ca

Hossein Ebrahimpour-komleh
Department of Electrical and Computer Engineering
University of Kashan
Kashan, Iran
ebrahimpour@kashanu.ac.ir

Shahryar Rahnamayan, SMIEEE
Nature Inspired Computational Intelligence (NICI) Lab
Department of Electrical, Computer, and Software Engineering
Ontario Tech University
Oshawa, Canada
Shahryar.Rahnamayan@ontariotechu.ca

***Abstract*—Crossover is the main genetic operator which influences the power of evolutionary algorithms. Among a variety of crossover operators, there has been a growing interest in multi-parent crossover operators in evolutionary computation. The main motivation of those schemes is establishing comprehensive collective collaboration of more than two chromosomes in the population to generate a new offspring. In this paper, a novel all-parent crossover operator called collective crossover for genetic algorithm is proposed. In this method, all individuals in the current population are involved in recombination part and one offspring is generated. The contribution of each individuals is defined based on its quality in terms of fitness value. The performance of the collective crossover operator is tested on CEC-2017 benchmark functions. The results revealed that the proposed crossover operator performs better when compared to well-known two-parent crossover operators including one-point and two-point crossovers. In addition, the differences between collective crossover and the other crossover operators are statistically significant for the most cases.**

***Index Terms*—Genetic algorithms, Crossover operator, Multi-parent crossover, Optimization, All-parent crossover.**

I. Introduction

Genetic algorithms are effective solvers for many global optimization problems. Crossover is one of the main operators in genetic algorithms (GAs) [1]. This operator in GAs enables the exchange of genetic information between parents to generate new offspring. Two parents are typically used for crossover operations in GAs in analogy to the sexual reproduction in biology. However, there are several studies in which multiple parents are used for crossover operators although there is no biological equivalence. Crossover operators using more than two parents have been successfully applied to the optimization problems [2]–[6].

Depending on the solution representation for the given optimization problems, different multi-parent crossover operators have been introduced in the literature. Two multi-parent crossover mechanism, namely gene scanning and diagonal crossover, were proposed in [2] . Gene scanning and diagonal crossover are general form of uniform and n-point crossover, respectively. These two methods are tested on benchmark functions (binary representation), a constraint satisfaction problem (order-based representation) and a constrained optimization problem (order-based representation). The results show that multi-parent crossover outperforms two-parent crossover for all benchmark functions and some problem instances of other problems.

Tsutsui and Jain [3] proposed binary GA that use multiple parents for crossover. They introduced two multi-parent crossover operators: multi-cut and seed crossover, each of which are the generalization of two-parent crossover. The performance of the operators were evaluated on the benchmark functions. Based on the results, the operators yield better performance.

A multi-parent simplex crossover was introduced for real-coded GAs in [7]. The crossover operator uses three or four parents depending on the dimension of the given problem. The offspring are created by using a uniform distribution with the property of a simplex.

Elyased et al. [4] presented a multi-parent crossover for real-coded GA. The crossover operator uses three parents that generates three offspring. It is responsible for both exploitation and exploration. In this study, a diversity operator was considered instead of mutation operator that prevents to get stuck in local optima. In addition, a memory that stores good candidate solutions are used in GA. The results indicate that the algorithm provides better performance when compared to the other methods.

In a recent study [6], the authors introduced a new multi-parent order crossover that has a reasonable computation time. As in many multi-parent crossover, this operator is an extension of two-parent crossover, i.e. more than two parents are used in recombination part. This operator creates only one offspring. According to experimental results, the proposed multi-parent crossover provide good performance on the travelling salesman problem and the berth allocation problem in terms of solution quality and the time.

As mentioned before, many different multi-parent crossover operators have been proposed that use multiple parents for recombination. Most of them use parents ranging from three to twenty that are smaller than the population size. In [5], the influence of the number of parents used were investigated. The results indicate that using a large number of parents provides better performance.

In this study, we introduce a novel all-parent crossover operator, called collective crossover, which uses all individuals in the population for GAs. The collective intelligence combines the genetic information of whole population, with more likely in the best individuals. In this crossover operator, only one child is generated based on gene scanning technique proposed in [2]. To evaluate the performance of the proposed method, a set of experiments are conducted using 29 benchmark functions proposed in the CEC-2017 special session on single objective real parameter optimization. The results of our experimental study demonstrate that the proposed crossover delivers good performance for majority benchmark problems.

The rest of the paper is organized as follows: Section II gives a brief information on GAs. Section III describes the proposed crossover operators for genetic algorithms. The experimental design and the results of this operator over a set of benchmark functions are presented in Section IV. Finally, Section V discusses the conclusions and highlights the promising future works.

II. Genetic Algorithm

Genetic Algorithm is a well-know biology-inspired algorithm to solve the optimization problems. The algorithm works in a repetitive procedure to find the optimal solution. At each iteration, some stochastic changes are applied to modify current candidate solutions to generate new ones. Similar to other population-based optimization algorithms, GA needs a population of chromosomes (i.e., individuals) to move toward the optimal solution(s) by evolving the population. For this purpose, the algorithm utilizes crossover and mutation operators to generate offspring from current parents in the population.

The offspring inherit the characteristics of the parents and will be added to the next generation if they have better fitness values compared to their parents. By this way, generations will be improved in terms of fitness values. Therefore, that is expected that the algorithm could find optimal solution after an appropriate number of iterations.

Two well-known crossovers that use two parents to generate offspring are one-point and two-point crossovers [6], [8]. In one-point crossover, a variable is selected randomly as the crossover point, then the all variables before that point of two parents are swapped to generate new off-springs. Using this operator, two new vectors are generated for more evaluation. Two-point crossover is a more generalized form of the one-point wherein two crossover points are selected randomly instead of one, then the middle parts of variables are swapped to generate offspring.

III. Proposed Collective Crossover

In this study, we propose a new crossover operator for GAs, namely collective crossover, which uses all individuals in the current population. Similar to a small society, contribution of individuals in the population can give more chance to find better candidate as a voting system. Accordingly, the proposed crossover utilize the whole population to get their characteristics based on their qualification in terms of fitness value to evolve the generations. The proposed crossover operator produces one child. The pseudo-code of collective crossover is given in Algorithm 1.

Algorithm 1 The pseudo-code of the proposed collective crossover

Input: The current population Pop_t
Output: The offspring population Pop_c

```
 1: Sort the population based on the fitness values
 2: for i:=1 to $N$ do
 3:     Create $N-i+1$ copies of $i^{th}$ individual in the
        population and insert into the mating pool
 4: end for
 5: for i:=1 to $n_c$ do
 6:     Generate a random permutation of genes ($perm$)
 7:     for j:=1 to $D$ do
 8:         Select a parent randomly from the mating pool ($p$)
 9:         $Pop_c[i].gene[perm(j)] = p.gene[perm(j)]$
10:         Remove one copy of the $p$ from mating pool
11:     end for
12: end for
```

Since the proposed crossover operator uses all individuals in the population, these individuals will be copied in the mating pool depending on their fitness values. In this method, better individuals contribute more copies to the mating pool. To create a mating pool (line 2-4 of Algorithm 1), all individuals in the population is sorted based on their fitness values. Then, the best individual in the population is copied as many as the population size (N); the second best individual is copied as many as $N-1$, and so on. Only one copy of the worst individual is created. As a result, there are $N*(N+1)/2$ individuals in the mating pool.

After creating a mating pool, a child is generated according to gene scanning technique [2]. Starting from a random gene position of all individuals in the mating pool, each gene is picked randomly from the corresponding gene of parents (line 5-12 of Algorithm 1); i.e. this operator change the whole genes (components) of the individual. Since mating pool includes more copies of better individuals, the child has more chance to inherits good genes from better parents. Thus, for each gene, a parent is selected randomly without replacement in mating pool to take the gene of offspring. Therefore, by selecting each parent from the mating pool, we decrease one chance (i.e. removing one copy) of selected parent to contribute in the next genes of the offspring. Thereafter, constructing the mating pool will be restarted for producing next offspring to give initial chances to all parents. Each time, the procedure of generating an offspring is performed based on a random order of genes. This prevents the same distribution of individuals for a specific gene each time. The proposed procedure will be repeated based on crossover rate (n_c) to generate offspring. Fig. 1 illustrates generating an offspring using collective crossover. In the figure, a sorted population, the initial constructed mating pool, and the generated offspring are presented.

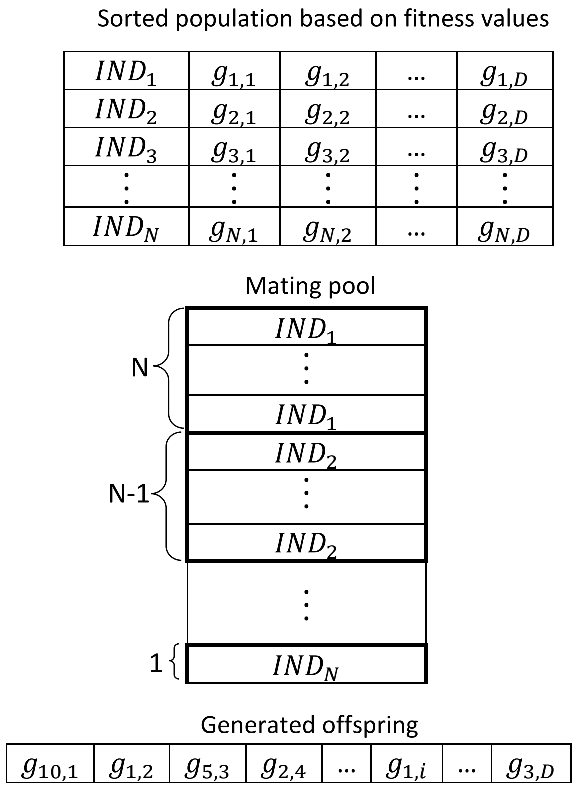
Sorted population based on fitness values
IND_1 $g_{1,1}$ $g_{1,2}$ ... $g_{1,D}$
IND_2 $g_{2,1}$ $g_{2,2}$ ... $g_{2,D}$
IND_3 $g_{3,1}$ $g_{3,2}$ ... $g_{3,D}$
IND_N $g_{N,1}$ $g_{N,2}$ ... $g_{N,D}$
Mating pool
N
IND_1
IND_1
N-1
IND_2
IND_2
1
IND_N
Generated offspring
$g_{10,1}$ $g_{1,2}$ $g_{5,3}$ $g_{2,4}$ ... $g_{1,i}$ ... $g_{3,D}$

Fig. 1. Generating an offspring using the proposed collective crossover. N is the population size.

The details of genetic algorithm proposed in this paper are as follows: A candidate solution is represented as a real-valued vector. The population is initialized randomly. The mating pool is generated according to fitness values. After applying collective crossover operator, the offspring population (Pop_c) is created and evaluated. To generate mutants (Pop_m), a number of individuals are chosen randomly from the current population (Pop_t). Mutation operator adds a random number drawn from a normal distribution to the selected gene. Then, Pop_t, Pop_c, and Pop_m are merged. The merged population is sorted and the best N individuals are selected for the next generation (line 8- 9 of Algorithm 2). The pseudo-code of the GA with collective crossover is given in Algorithm 2.

IV. Experiments

A. Experimental Settings

To evaluate the performance of the proposed method, 29 benchmark functions proposed in the CEC-2017 special session on single objective real parameter optimization were used [9]. These benchmark functions have different characteristics: unimodal functions (F1-F2), simple multimodal functions (F3-F9), hybrid functions (F10-F19) and composite functions (F20-F29). The details of the functions is presented in Table I.

TABLE I
Summary of the CEC-2017 test functions. M is the number of basic functions to construct the components of hybrid or composition functions.

No.	Functions	Type
F1	Shifted and Rotated Bent Cigar Function	Unimodal Functions
F2	Shifted and Rotated Zakharov Function	
F3	Shifted and Rotated Rosenbrock's Function	Simple Multimodal Functions
F4	Shifted and Rotated Rastrigin's Function	
F5	Shifted and Rotated Expanded Scaffer's F6 Function	
F6	Shifted and Rotated Lunacek Bi_Rastrigin Function	
F7	Shifted and Rotated Non-Continuous Rastrigin's Function	
F8	Shifted and Rotated Levy Function	
F9	Shifted and Rotated Schwefel's Function	
F10	Hybrid Function 1 (M=3)	Hybrid Functions
F11	Hybrid Function 2 (M=3)	
F12	Hybrid Function 3 (M=3)	
F13	Hybrid Function 4 (M=4)	
F14	Hybrid Function 5 (M=4)	
F15	Hybrid Function 6 (M=4)	
F16	Hybrid Function 6 (M=5)	
F17	Hybrid Function 6 (M=5)	
F18	Hybrid Function 6 (M=5)	
F19	Hybrid Function 6 (M=6)	
F20	Composition Function 1 (M=3)	Composition Functions
F21	Composition Function 2 (M=3)	
F22	Composition Function 3 (M=4)	
F23	Composition Function 4 (M=4)	
F24	Composition Function 5 (M=5)	
F25	Composition Function 6 (M=5)	
F26	Composition Function 7 (M=6)	
F27	Composition Function 8 (M=6)	
F28	Composition Function 9 (M=3)	
F29	Composition Function 10 (M=3)	
Search Range: $[-100, 100]^D$		

The performance of the proposed crossover operator was compared with two well-known crossover operators, namely one-point and two-point crossover operators [10]. In this study, the public MATLAB implementation of the genetic algorithm (GA) in [11] was adapted. The roulette wheel selection was used as the parent selection method for the GA with one-point and two-point crossover operators.

The parameter settings for all GAs are given as follows:

- The maximum number of generations is set to $30 * D$, where D is the number of dimensions.
- Population size (N) is set to 100.
- Crossover percentage (p_c) is set to 0.7.
- The number of offspring (n_c) is $2 * round(p_c * N/2)$.
- Crossover probability is set to 1.0.
- Mutation percentage (p_m) is set to 0.3.

Algorithm 2 The pseudo-code of the GA with the proposed collective crossover

```
1: $t = 0$
2: Create an initial population ($Pop_t$) and evaluate $Pop_t$
3: **while** (termination criteria not satisfied) **do**
4:     Create offspring ($Pop_c$) by using *Algorithm 1* and evaluate $Pop_c$
5:     Select individuals randomly from the $Pop_t$ for mutation
6:     Apply mutation to create mutants ($Pop_m$) and evaluate $Pop_m$
7:     Merge three populations ($MPop_t = Pop_t \cup Pop_c \cup Pop_m$)
8:     Sort $MPop_t$ based on the fitness values ($sort(MPop_t)$)
9:     Select the best $N$ individuals from the sorted merged population ($Pop_{t+1} = selectBestIndividual(MPop_t)$)
10:    $t = t + 1$
11: **end while**
```

- The number of mutants (n_m) is $round(p_m * N)$.
- The mutation rate is set to 0.1.

The performance of the methods is compared based on the function error values: $f(x) - f(x^*)$ where x is the best solution obtained and x^* is the global optimum of the corresponding benchmark function. In the tables, the results are reported in terms of averaged error values over 30 independent runs. Each row except for the last row indicates the performance of all the methods for the corresponding benchmark function. In the columns of the tables, the averaged error value of each method is presented. Additionally, in all the tables, the best performing method is marked in bold.

The Wilcoxon signed-ranks test at a 95% confidence interval was performed for the statistical comparison of crossover operators. In order to summarize the statistical comparison results, the win/tie/lose (*w/t/l*) values are provided in the last row of the result tables.

B. Results and Discussion

The average error values generated by different crossover operators on the benchmark functions with 30, 50, and 100 dimensions are provided in Tables II, III, and IV, respectively. These tables also present the *w/t/l* values for each dimension. As shown in the Table II, the proposed crossover could outperform two other competitors on most of the functions. The collective crossover could reach lower fitness values on 17 out of 29 functions compared to other crossover competitors. In addition, on remaining functions, all methods have same efficiency to approach the optimal solution. The results also indicate that the difference between the proposed method and other methods is significant on most of the winner functions. For instance, the error of algorithm to find the optimal solution by using collective crossover for F8 is 84.76 while two other operators could find the solution with error 1247.88 and 1036.69, respectively. There are several other functions among benchmark functions with similar remarkable difference between the result of collective operator and other operators such as F12, F14, F17.

Similar results in the Table III can be observed for experiments on dimension 50. The collective crossover gets better fitness values on 20 out of 29 functions compared to one-point crossover while they have same performance on 8 out of 29 functions. The proposed operator was not able to find a better solution on only one function. The comparison results between collective and two-point crossovers on dimension 50 are almost the same. The number of winner functions for collective crossover is 19 out of 29 while the same error has been obtained on the remaining functions, except F2. The point worth mentioning is that by increasing the dimension from 30 to 50, the efficiency of collective operator is not affected according to the number of winner functions. The difference between the resulted error of collective operator and competitor crossovers is more considerable in higher dimensions. The same observation can be found on the results of dimension 100 presented in Table IV. The collective operator outperforms the one-point crossover on 17 out of 29 functions while the number of tie and lose functions are 10 and 2, respectively. The obtained results are almost same on comparison between the collective and two-point crossovers except one more winner function. The overall results of mentioned experiments are indicated in Table V. In the proposed crossover, intelligence of whole population contribute to generate a candidate solution with a better fitness value. Furthermore, the higher probability of selecting better candidate solutions leads to inheriting the characteristic of those solutions in generated offspring. The provided results indicate that the positive effect of the collective intelligence directs the population toward to more promising regions of the search space.

We also perform a rank test based on Friedman test for the statistical comparison. In Friedman test, each method is ranked for each problem. Then, the total rank (score) for the corresponding method is calculated. In this method, the smaller the score, the better the performance. The average ranking based on the Friedman test of each method is illustrated in Fig. 2 for each dimension. It can be seen that the collective crossover has higher rank comparing to other methods on all experimental dimensions. As mentioned previously, the collective crossover performs better to reach optimal solutions in higher dimension comparing to its competitors. The average rank based on Friedman test is an indicative for this point. The presented plot also provides a comparison of the methods on different dimensions. Accordingly, the collective crossover has the highest rank on dimension 100 comparing to other dimensions and other methods.

TABLE II
ERROR VALUES GENERATED BY THREE CROSSOVER OPERATORS ON 30D FUNCTIONS. THE LOWER ERROR VALUES ARE HIGHLIGHTED. LAST ROW REPRESENTS THE STATISTICAL TEST RESULTS.

Function	Collective Crossover	One-point Crossover	Two-point Crossover
F1	**3.79E+13**	4.12E+13	4.05E+13
F2	**2.18E+10**	2.42E+10	2.80E+10
F3	**1.15E+08**	1.27E+08	1.31E+08
F4	**6.61E+07**	1.19E+08	9.90E+07
F5	2.35E+06	**2.23E+06**	2.43E+06
F6	**1.50E+08**	1.86E+08	1.82E+08
F7	**5.75E+07**	1.15E+08	9.65E+07
F8	**8.48E+07**	1.25E+09	1.04E+09
F9	**2.91E+09**	3.45E+09	3.15E+09
F10	**1.82E+08**	2.41E+08	2.34E+08
F11	**2.12E+12**	3.13E+12	2.85E+12
F12	**2.32E+10**	2.22E+11	1.21E+11
F13	**4.70E+11**	5.63E+11	8.01E+11
F14	**8.67E+09**	1.72E+10	1.45E+10
F15	**8.20E+08**	1.17E+09	1.08E+09
F16	**2.73E+08**	6.07E+08	4.60E+08
F17	**9.18E+11**	1.24E+12	1.30E+12
F18	**1.06E+10**	1.29E+10	1.94E+10
F19	**4.03E+08**	5.05E+08	4.79E+08
F20	**2.65E+08**	3.10E+08	3.04E+08
F21	**7.18E+08**	1.85E+09	9.31E+08
F22	**4.07E+08**	4.88E+08	4.56E+08
F23	**4.99E+08**	5.77E+08	5.45E+08
F24	**3.99E+08**	4.05E+08	4.05E+08
F25	**1.60E+09**	2.42E+09	2.26E+09
F26	**5.28E+08**	5.33E+08	5.29E+08
F27	4.75E+08	4.71E+08	**4.70E+08**
F28	**6.83E+08**	9.12E+08	8.70E+08
F29	**1.24E+10**	1.68E+10	1.62E+10
w/t/l	-	*17/12/0*	*17/12/0*

TABLE III
ERROR VALUES GENERATED BY THREE CROSSOVER OPERATORS ON 50D FUNCTIONS. THE LOWER ERROR VALUES ARE HIGHLIGHTED. LAST ROW REPRESENTS THE STATISTICAL TEST RESULTS.

Function	Collective Crossover	One-point Crossover	Two-point Crossover
F1	**3.61E+14**	4.23E+14	4.12E+14
F2	8.91E+10	**6.48E+10**	6.50E+10
F3	**2.44E+08**	3.28E+08	3.02E+08
F4	**2.07E+08**	3.48E+08	3.32E+08
F5	**5.43E+06**	7.59E+06	6.66E+06
F6	**3.73E+08**	4.72E+08	4.53E+08
F7	**1.76E+08**	3.41E+08	3.00E+08
F8	**5.61E+08**	1.45E+10	8.68E+09
F9	**7.06E+09**	7.59E+09	7.48E+09
F10	**5.37E+08**	6.10E+08	6.56E+08
F11	**2.34E+13**	2.57E+13	2.83E+13
F12	**5.13E+10**	6.39E+10	7.57E+10
F13	**1.22E+12**	1.97E+12	1.72E+12
F14	**1.57E+10**	2.19E+10	1.85E+10
F15	**1.27E+09**	1.84E+09	1.72E+09
F16	**1.05E+09**	1.57E+09	1.35E+09
F17	**4.02E+12**	5.19E+12	6.35E+12
F18	1.81E+10	**1.46E+10**	1.74E+10
F19	**9.93E+08**	1.06E+09	1.13E+09
F20	**4.13E+08**	5.19E+08	4.90E+08
F21	**7.39E+09**	8.31E+09	8.48E+09
F22	**6.21E+08**	8.34E+08	7.80E+08
F23	**7.62E+08**	9.23E+08	8.50E+08
F24	**6.30E+08**	6.74E+08	6.73E+08
F25	**2.83E+09**	5.12E+09	4.77E+09
F26	**7.02E+08**	8.15E+08	7.80E+08
F27	6.31E+08	**6.30E+08**	6.40E+08
F28	**8.23E+08**	1.37E+09	1.38E+09
F29	**1.07E+12**	1.41E+12	1.36E+12
w/t/l	-	*20/8/1*	*19/9/1*

V. CONCLUSION REMARKS

In this paper, a new crossover operator named collective crossover is proposed for evolutionary algorithms. The proposed operator works based on the collaboration of individuals in whole population. Unlike many existing crossover operators which an offspring is generated using a limited number of candidate solutions, the collective crossover gathers the intelligence of whole population with giving more probability to the best individuals. Similar to a small society, a collaboration on whole population leads to generate new candidates with better fitness value. The results of conducted experiments on CEC-2017 benchmark functions indicate the efficiency of the GA with utilizing the collective crossover comparing to two well-known crossover operators: one-point and two-point crossovers. In addition, the results on dimensions 30, 50, and 100 provide hints that increasing the dimension preserves the effectiveness of the operator even with better rank. Although the proposed method requires longer computational time, it provides promising results.

As the future works, we will compare with other multi-parent crossover operators, in particular gene scanning crossover operators. We will also perform a comprehensive analysis of the proposed operator. Another future work can be utilizing the operator on large-scale optimization problems.

REFERENCES

[1] G. Pavai and T. V. Geetha, “A survey on crossover operators,” *ACM Comput. Surv.*, vol. 49, no. 4, Dec. 2016. [Online]. Available: https://doi.org/10.1145/3009966

[2] A. E. Eiben, P.-E. Raué, and Z. Ruttkay, “Genetic algorithms with multi-parent recombination,” in *Parallel Problem Solving from Nature (PPSN)*, 1994.

TABLE IV
ERROR VALUES GENERATED BY THREE CROSSOVER OPERATORS ON 100D FUNCTIONS. THE LOWER ERROR VALUES ARE HIGHLIGHTED. LAST ROW REPRESENTS THE STATISTICAL TEST RESULTS.

Function	Collective Crossover	One-point Crossover	Two-point Crossover
F1	**3.20E+15**	3.66E+15	3.49E+15
F2	3.65E+11	2.83E+11	**2.82E+11**
F3	**7.25E+08**	9.78E+08	9.50E+08
F4	**7.10E+08**	1.18E+09	1.05E+09
F5	**1.26E+07**	3.19E+07	2.72E+07
F6	**1.01E+09**	1.51E+09	1.55E+09
F7	**6.66E+08**	1.22E+09	1.04E+09
F8	**2.51E+09**	6.97E+10	5.53E+10
F9	**2.20E+10**	2.32E+10	2.31E+10
F10	4.39E+09	3.54E+09	**3.51E+09**
F11	**3.95E+14**	4.51E+14	4.27E+14
F12	**2.91E+11**	6.48E+11	3.27E+11
F13	5.14E+12	**4.92E+12**	5.17E+12
F14	**1.98E+10**	2.41E+11	7.07E+10
F15	**3.20E+09**	4.69E+09	4.49E+09
F16	**2.80E+09**	3.69E+09	3.89E+09
F17	**7.57E+12**	8.38E+12	8.75E+12
F18	**1.89E+10**	2.02E+10	3.27E+10
F19	**2.59E+09**	3.29E+09	3.08E+09
F20	**9.24E+08**	1.40E+09	1.30E+09
F21	**2.31E+10**	2.41E+10	2.46E+10
F22	**9.77E+08**	1.52E+09	1.42E+09
F23	**1.43E+09**	2.24E+09	2.13E+09
F24	**1.38E+09**	1.54E+09	1.61E+09
F25	**9.64E+09**	1.79E+10	1.54E+10
F26	**9.06E+08**	1.22E+09	1.11E+09
F27	1.29E+09	1.25E+09	**1.23E+09**
F28	**3.01E+09**	5.01E+09	4.89E+09
F29	**3.33E+11**	4.06E+11	3,65E+11
w/t/l	-	*17/10/2*	*18/9/2*

TABLE V
THE OVERALL COMPARISON ($w/t/l$) ON THE PROPOSED COLLECTIVE CROSSOVER AND TOW-PARENT CROSSOVERS.

	30D	50D	100D
One-point Crossover	17/12/0	20/8/1	17/10/2
Two-point Crossover	17/12/0	19/9/1	18/9/2

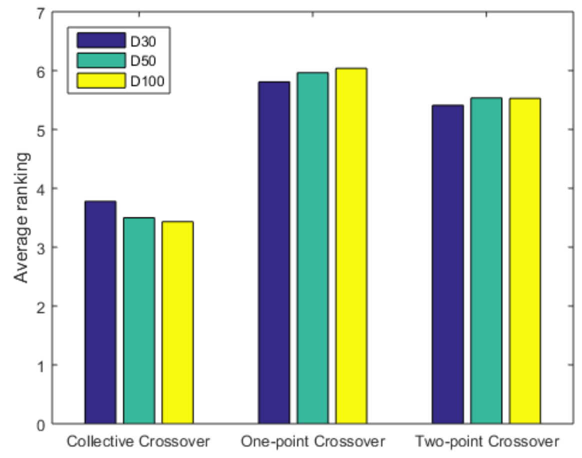

Fig. 2. Average ranking based on the Friedman test for each method and dimension.

[3] S. Tsutsui and L. C. Jain, "On the effect of multi-parents recombination in binary coded genetic algorithms," in *1998 Second International Conference. Knowledge-Based Intelligent Electronic Systems. Proceedings KES'98 (Cat. No.98EX111)*, vol. 3, April 1998, pp. 155–160 vol.3.

[4] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new genetic algorithm for solving optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 57 – 69, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197613001875

[5] S. P. T. P. Phyu and G. Srijuntongsiri, "Effect of the number of parents on the performance of multi-parent genetic algorithm," in *2016 11th International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, Nov 2016, pp. 1–6.

[6] A. Arram and M. Ayob, "A novel multi-parent order crossover in genetic algorithm for combinatorial optimization problems," *Computers & Industrial Engineering*, vol. 133, pp. 267 – 274, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835219302773

[7] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, ser. GECCO'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 657–664.

[8] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, and K. Leong, "Crossover and mutation operators of genetic algorithms," *International journal of machine learning and computing*, vol. 7, no. 1, pp. 9–12, 2017.

[9] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang, and B. Y. Qu, "Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Nanyang Technological University (Singapore), Jordan University of Science and Technology (Jordan) and Zhengzhou University (Zhengzhou China), Tech. Rep., 2016.

[10] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Springer Publishing Company, Incorporated, 2015.

[11] Yarpiz, "Binary and real-coded genetic algorithms in matlab-yarpiz," https://yarpiz.com/23/ypea101-genetic-algorithms, accessed: July 15, 2019.