On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: a first study on the TSP

E. Osaba, R. Carballedo, F. Diaz, E. Onieva, P. Lopez, A. Perallos

***Abstract*—Combinatorial optimization is a widely studied field within artificial intelligence. There are many problems of this type, and many techniques applied to them can be found in the literature. Especially, population techniques have received much attention in this area, being genetic algorithms (GA) the most famous ones. Although throughout history many studies on GAs have been performed, there is still no study like the presented in this work. In this paper, a study on the influence of using heuristic initialization functions in genetic algorithms (GA) applied to combinatorial optimization problems is performed. Being the first phase of this research, the study is conducted using one of the best known problems in combinatorial optimization: the traveling salesman problem. Three different experimentations are carried out, using three different heuristic initialization functions. Additionally, for each experiment four versions of a GA have been developed for the comparison. Each of these variant differs in the initialization phase. The results obtained by each GA are compared to determine the influence of the use of heuristic functions for the initialization of the population.**

Keywords*—Genetic algorithm, Combinatorial optimization, Meta-heuristic, Initialization, Traveling Salesman Problem.

I. Introduction

Due to its real-world application, combinatorial optimization has been one of the most studied fields of artificial intelligence. Problems of this type are the focus of many research studies every year [1, 2]. Some classic examples of such problems are the Traveling Salesman Problem (TSP) [3], the Capacitated Vehicle Routing Problem [4], or the Job-shop Scheduling Problem [5]. The interest of these problems lies on their complexity and applicability to real life. Being NP-Hard [6], a large number of techniques applied to these problems can be found in the literature. Some of the most commonly used are the simulated annealing [7], the tabu search [8] and genetic algorithms (GA) [9, 10].

In some studies, combinatorial optimization problems are used as benchmarking problems. In other words, they are used to compare the performance of different techniques. Some papers of this sort are [11–13]. In this case, as can be read in [14], to make fair and rigorous comparisons, the best way to implement the algorithms is using neutral functions, i.e., functions that do not use specific knowledge of the problem in the optimization process (optimization heuristics), except to assure the generation of feasible solutions that verify the hard constraints of the problem.

Besides serving as a method to validate the performance of metaheuristic algorithms, there are many research studies whose sole purpose is the resolution of combinatorial optimization problems. This is due to the practical application of these problems in real life: for example, in transportation and industrial manufacturing. Some recent examples of these works are [15, 16]. In this kind of studies, with the aim of exploiting and exploring the solution space more exhaustively, heuristic functions are used. Using the TSP and a GA as example, the nearest neighbor (NN) [17] is one of these functions. This operator uses the distance between nodes with the aim of create better solutions. Some other operators of this type can be the sequential constructive crossover [18], or the greedy sub-tour mutation [19].

In this paper the first stage of a study on the influence of using heuristic initialization functions in GAs applied to combinatorial optimization problems is performed. Nowadays, population algorithms have become one of the most successful meta-heuristics for solving combinatorial optimization problems. Thanks to their robustness and their adaptability to a wide variety of problems, diverse population algorithms have been proposed along the history, such as the GAs, evolutionary algorithms [20], distributed population algorithms [21], or artificial bee colony algorithms [22]. These algorithms work with a population of solutions, which evolve along the execution. In many cases, these initial solutions are randomly created. In some other cases they are created using heuristic functions. Thus, the solutions go through a preliminary optimization process, increasing the likelihood of obtaining better solutions at the end of the execution of the algorithm. This heuristic initialization is especially used in problem solving, and not so much in meta-heuristic performance comparisons, where the use of problem heuristics can limit the generality of the conclusions with respect to problems of different type.

Among all the population techniques, the GA has received most attention. Its effectiveness for solving optimization problems has been proven on many occasions. Some recent examples are [23–25]. In addition, there are many studies in the literature focused on the theoretical and practical analysis of GAs. This kind of research analyzes, for example, behavioural characteristics of the algorithm, as the convergence [26]; the efficiency of certain phases of the GA, such as crossover [14, 27] or mutation [28, 29]; or the influence of adapting some parameters, as the crossover and mutation probability

Deusto Instute of Technology, University of Deusto,
Av. Universidades, 24, Bilbao, Spain
Email: {e.osaba, roberto.carballedo, fernando.diaz}@deusto.es
{enrique.onieva, p.lopez, perallos}@deusto.es

[30, 31].

Even so, although there are some works focused on other topics related to the population [32, 33], there is still no study that analyzes the effect on initializing the population of a GA with an elitist (or semi-elitist) survivors selection criterion applied to combinatorial optimization problems using heuristic functions. The absence of a study of this type has been the motivation for the research presented in this paper.

The rest of the paper is organized as follows: in Section II the description of the experimentation is presented. In Section III, the experiments performed are shown, and the results are analyzed. Finally, this paper is completed with the conclusions of the study and further work (Section IV).

II. Description of the experimentation

In this section the conducted experimentation is described. This fact has a great importance for the authors of this paper, since one of the main objectives of this study is to facilitate its replicability.

Being the first phase of this research, the study is conducted using one of the most studied problems in combinatorial optimization: the TSP. This problem has not been used to demonstrate that the GA is a good alternative to solve it, something already proven. This problem is used because it is well known, it is simple to implement and understand, and it is easily replicable.

In this study three different experiments are carried out, each one with a different initialization function. The investigated heuristics are the following:

- Nearest Neighbor (NN): This classic function begins by selecting a random node as start node. Then iteratively, the nearest node to the last node added is added to the partial path. This function is widely used by the researchers annually [34, 35].
- Insertion (In): This function starts the process with a partial path consisting of one randomly selected arc. From this point and iteratively, the nearest node to any node in the partial path is obtained. This edge is inserted into the path at the position that involves lowest cost increase. As the NN, this function is also used in many works [36–38].
- Solomon Heuristic I1 (I1): This function is a modification of the heuristic proposed by Solomon in 1987 for the Vehicle Routing Problem with Time Windows [39]. This function starts with a partial path consisting of two randomly selected nodes. Then, iteratively, for each node(which is not yet in the path, the cost for inserting it at each of the possible positions on the partial tour is calculated. The node, that induces the smallest increase is inserted into the path on the optimal position.

Additionally, for each experiment four different versions of the GA have been developed for the comparison. Each of these variants has been called GA_α, where α is the percentage of the population created using heuristic initialization functions. In this paper α takes four different values: 100, 50, 10, and 0.

In this way, the objective of this study is to compare the performance of these four algorithms applied to the TSP, with the aim of checking the effect of initializing the population using heuristic functions. In addition, it will be examined which is the best parametrization. The structure used for the GAs is the represented in Algorithm 1, and it is considered the conventional one.

Algorithm 1: Pseudocode of GAs

```
Initialization of population
repeat
    Parents selection process
    Crossover phase
    Mutation phase
    Survivor selection process
until termination criterion reached;
Return the fitness of the best individual found
```

In order to obtain valid and objective conclusions, the good practices proposed in [40] have been followed to develop the GAs. These practices dictate that the most appropriate way to determine the efficiency of a particular operator is to use techniques with neutral functions, so that the optimization capacity of the new operator is not "tarnished". This way, as has been said, the only difference between the different GAs implemented is the initialization process.

Regarding the parameters of the algorithms, for all the GAs, the population is composed by 48 individuals. All the solutions are encoded following the path representation [41]. The parametrization of the GAs has been made based on the concepts outlined in many previous studies [42–44]. According to these research works, the crossover is considered the main operator of genetic algorithms. On the other hand, the mutation is a secondary operation. In this way, all the GAs have a crossover probability (p_c) of 90%, and a mutation probability of 10%.

In relation to the parents selection criteria, first, each individual of the population is selected as parent with a probability equal to p_c. If one individual is selected for the recombination, the other parent is selected randomly. Regarding the survivor function, a 50% elitist-random function has been used. About the termination criterion, the execution of each technique finishes when there are $n+\sum_{k=1}^{n} k$ generations without improvements in the overall fitness of the population, where n is the size of the problem. As crossover and mutation functions, the well-known order crossover [45] and 2-opt [46] operators are used, respectively. These functions have been widely used in the literature [47–50]

III. Experimentation

In this section the performed experimentation is shown. All the tests were performed on an Intel Core i5 2410 laptop, with 2.30 GHz and 4 GB of RAM. Java was used as programming language. For each run, the total average, and standard deviation are shown. The average runtime is also displayed, in seconds. Each experiment is repeated 50 times, and all instances have been obtained from the TSPLIB Benchmark [51]. The name of each TSP instance has a number

TABLE I. RESULTS OF THE FOUR GAS USING THE NN FUNCTION

Instance		GA_{100}			GA_{50}			GA_{10}			GA_0		
Name	Optima	Avg.	S. dev.	time	Avg.	S. dev.	time	Avg.	S. dev.	time	Avg.	S. dev.	time
Oliver30	420	424.2	4.15	0.39	**422.3**	3.37	0.40	422.4	2.64	0.48	423.7	7.50	0.60
Eilon50	425	434.8	4.91	2.55	436.1	4.83	2.37	**433.6**	6.15	2.86	442.9	7.32	3.20
Berlin52	7542	7730.4	206.12	1.71	**7653.5**	171.44	1.84	7788.0	209.67	1.97	7951.7	190.94	1.69
St70	675	695.0	9.17	3.45	**689.4**	7.94	3.85	693.8	9.00	3.98	712.3	17.10	6.51
Eilon75	535	560.4	10.24	6.74	**558.3**	6.25	6.20	562.1	7.34	5.40	573.0	8.91	7.95
Eil76	538	560.5	7.63	6.46	**558.7**	6.36	5.36	561.0	7.63	5.66	574.6	13.00	10.25
KroA100	21282	21776.8	288.18	14.35	**21625.7**	283.21	12.82	21643.4	294.92	11.36	22140.1	635.51	17.12
KroB100	22140	22636.9	212.43	13.20	22695.3	255.07	13.16	**22612.1**	298.06	13.58	22938.2	327.19	20.12
KroC100	20749	21553.1	253.66	13.45	**21473.6**	256.97	14.02	21492.3	334.42	12.78	21915.6	479.65	18.18
Eil101	629	665.8	9.09	9.16	**659.0**	10.37	10.50	660.6	9.06	9.47	690.3	14.15	13.91
Pr107	44303	44522.8	119.05	13.17	**44518.1**	109.31	16.33	44524.7	215.57	13.73	47274.9	1621.38	18.42
Pr124	59030	**59903.9**	520.85	28.44	59997.2	385.3	27.16	60263.9	434.84	23.96	60521.7	998.58	28.46
Pr136	96772	102660.1	869.51	36.43	102707.8	681.59	38.14	103636.2	1212.20	35.91	**102610.6**	1260.51	40.12
Pr144	58537	59433.0	864.50	41.25	**59313.1**	821.65	43.20	59412.0	884.68	39.84	60182.6	1440.60	46.23
Pr152	73682	75238.8	772.26	50.15	**75068.6**	661.82	49.39	75230.1	874.06	49.11	76467.3	1407.02	54.61

TABLE II. RESULTS OF THE FOUR GAS USING THE IN FUNCTION

Instance		GA_{100}			GA_{50}			GA_{10}			GA_0		
Name	Optima	Avg.	S. dev.	time	Avg.	S. dev.	time	Avg.	S. dev.	time	Avg.	S. dev.	time
Oliver30	420	425.1	4.42	0.22	**423.7**	3.84	0.31	423.8	4.77	0.26	**423.7**	7.50	0.60
Eilon50	425	443.7	5.60	0.76	**442.5**	4.58	0.92	447.8	8.31	1.42	442.9	7.32	3.20
Berlin52	7542	7971.6	155.28	3.39	**7928.3**	189.42	2.31	7961.1	178.64	2.48	7951.7	190.94	1.69
St70	675	708.1	10.12	3.77	709.8	13.84	4.06	**707.5**	12.53	4.75	712.3	17.10	6.51
Eilon75	535	573.3	11.56	4.45	**571.9**	8.57	6.46	580.7	8.63	6.65	573.0	8.91	7.95
Eil76	538	571.0	7.53	4.21	**569.1**	8.76	6.46	581.5	11.34	5.90	574.6	13.00	10.25
KroA100	21282	22008.2	227.52	15.16	**21903.5**	365.26	15.32	22274.2	292.43	12.15	22140.1	635.51	17.12
KroB100	22140	22931.0	367.03	16.23	**22854.1**	425.00	13.80	23057.8	445.38	13.83	22938.2	327.19	20.12
KroC100	20749	22148.2	397.55	13.01	**22136.6**	483.04	12.95	22283.9	460.65	16.42	22815.6	479.65	18.18
Eil101	629	**671.0**	7.45	9.31	672.5	7.47	10.80	685.0	9.94	9.83	690.3	14.15	13.91
Pr107	44303	**45150.4**	289.47	18.17	45544.0	742.16	14.34	46062.8	1504.2	19.11	47274.9	1621.38	18.42
Pr124	59030	60238.4	2212.12	20.13	**59771.4**	543.21	22.46	60501.5	771.33	21.21	60521.7	998.58	28.46
Pr136	96772	100865.5	799.51	26.15	**100317.9**	933.23	28.12	101298.7	806.71	29.72	102610.6	1260.51	40.12
Pr144	58537	**59702.9**	353.85	33.80	59737.2	385.22	31.25	59988.3	836.00	32.15	60182.6	1440.60	46.23
Pr152	73682	76090.3	1373.97	58.16	**75667.7**	801.09	53.12	77236.3	1280.45	52.15	76467.3	1407.02	54.61

that displays the number of nodes it has. Table I shows the results achieved with the NN function. Table II displays the outcomes of the In operator. Finally, Table III shows the results obtained with I1.

Some conclusions can be drawn analyzing these results. First, it can be seen how the use of heuristic initialization functions helps to get better results, since the GA_0 obtained worse results than the other alternatives in the 62.22% of the cases (28 out of 45). Anyway, the obtaining of this conclusion is logical, given that the individuals of the other GA versions go through an optimization process before the execution of the algorithm. For this reason, it is easier for these GAs to obtain better results.

The second reflection that can be done from these outcomes is the following. Having seen the influence on the results of using heuristic initialization functions, the next step is to determine which is the suitable number of individuals initialized with them. Knowing the advantages of heuristic initialization, one could make the mistake of believing that the more individuals initialized, the more probability to get a better final solution. Nonetheless, as can be seen in the results obtained, the key is to maintain a balance between individuals initialized by heuristic functions, and individuals generated randomly. In relation to Table I, it can be observed that the GA that obtains best results is the GA_{50}. Although sometimes the differences are not very wide, it can be seen a clear trend of improvement. In fact, the GA_{50} performs better in 73.33% (11 out of 15) of the instances. In the case of the In function, Table II shows that the results are similar to those obtained for the function NN. In this occasion, GA_{50} outperforms the other

TABLE III. RESULTS OF THE FOUR GAS USING THE I1 FUNCTION

Instance		GA_{100}			GA_{50}			GA_{10}			GA_0		
Name	Optima	Avg.	S. dev.	time	Avg.	S. dev.	time	Avg.	S. dev.	time	Avg.	S. dev.	time
Oliver30	420	425.4	4.24	0.25	**422.4**	3.17	0.22	425.3	3.59	0.44	423.7	7.50	0.60
Eilon50	425	439.5	5.73	1.79	**437.5**	3.94	2.15	441.7	6.45	2.61	442.9	7.32	3.20
Berlin52	7542	7868.2	183.46	2.06	**7808.6**	182.30	2.37	7993.7	264.06	2.36	7951.7	190.94	1.69
St70	675	**690.4**	9.08	5.15	691.9	9.01	5.13	701.3	10.74	4.29	712.3	17.10	6.51
Eilon75	535	562.3	4.73	3.83	**560.8**	8.50	4.12	571.2	8.76	5.02	573.0	8.91	7.95
Eil76	538	562.6	5.05	2.66	**560.9**	6.61	4.01	570.9	9.68	4.31	574.6	13.00	10.25
KroA100	21282	**21894.3**	437.09	13.26	22041.2	419.33	14.98	22358.9	621.58	16.43	22140.1	635.51	17.12
KroB100	22140	23059.6	266.06	11.98	**22970.1**	284.20	14.33	23398.2	250.89	13.75	22938.2	327.19	20.12
KroC100	20749	22121.1	437.34	13.40	22037.8	229.91	12.17	22457.3	679.29	12.83	**21915.6**	479.65	18.18
Eil101	629	672.4	10.17	9.23	**668.6**	6.73	9.19	674.4	10.38	12.52	690.3	14.15	13.91
Pr107	44303	45219.3	346.81	12.30	**45110.1**	270.92	13.10	45239.3	1104.25	14.80	47274.9	1621.38	18.42
Pr124	59030	60017.7	432.00	19.82	**59883.0**	570.56	18.66	60391.0	776.74	19.42	60521.7	998.58	28.46
Pr136	96772	99404.1	1498.80	28.31	**98839.9**	573.93	31.20	99406.4	1098.84	32.15	102610.6	1260.51	40.12
Pr144	58537	60334.0	1057.50	37.54	60278.1	1045.31	36.12	60627.4	1576.61	37.25	**60182.6**	1440.60	46.23
Pr152	73682	**75821.1**	865.04	54.35	76205.7	900.70	53.42	76204.1	914.78	54.26	76467.3	1407.02	54.61

GAs also in the 73.33% of the instances. Finally, in Table III can be observed how this trend also occurs for the function I1. In this case, GA_{50} obtains better outcomes in 66.66% of the instances.

These results can be explained as follows. As already mentioned, the use of heuristic initialization functions increases the quality of the results. However, overuse of these functions carries a decrease in the exploration capability of the technique. Thus, the population could be trapped in local optima very quickly. On the other hand, maintaining a balance as presented in the GA_{50}, the search can benefit from the previous optimization applied to the 50% of the individuals. Meanwhile, the exploration capacity of the technique is maintained by the remaining 50% of the population, which is generated randomly.

Regarding the runtimes, in all the tables can be seen that they are similar for GA_{100}, GA_{50}, and GGA_{10}, the algorithms using heuristic initialization functions. This fact is repeated for the three experiments performed, so it can be concluded that a higher (or lower) proportion of individuals created by heuristic functions does not influence runtimes. Furthermore, compared with the other three GAs, the execution times are higher for the algorithm which initializes the population randomly, i.e. GA_0. This occurs because the latter algorithm needs more time to reach convergence. This is a logical behavior, since, generally, randomly generated individuals have a very bad fitness. For this reason, these individuals need more generations to reach a local optima. This leads to a slower convergence of the algorithm, and therefore, to a longer execution time.

In Table IV the results of overall ranking calculated using the well-known statistical Friedman's test are summarized. It is noteworthy that the smaller the score, the better the ranking. These results also demosntrate that GA_{50} is the best GA for the three experimentations performed. Additionally, in order to check if there are statistical differences between the different GAs, the value of X_r^2 is also depicted in Table IV. This value has been obtained using the following formula:

$$X_r^2 = \frac{12}{HK(K+1)} \sum HRc^2 - 3H(K+1)$$

Being H the number of instances (15), K the number of techniques (4), and Rc the value of the Friendman's test score. The critical point in a χ^2 distribution with 3 degrees of freedon is 12.838. Besides, the confidence interval has been stated at the 99'5% confidence level. The X_r^2 values obtained for all the tests are higher than 12.838. For this reason, it can be said that there are significant differences between the four GAs for the three experimentations performed.

TABLE IV. RESULTS OF FRIEDMAN'S TEST

Function	GA_{100}	GA_{50}	GA_{10}	GA_0	X_r^2
NN	2.46	**1.40**	2.30	3.70	23.56
In	2.13	**1.36**	3.30	3.16	23.02
I1	2.13	**1.40**	3.26	3.20	21.80

Finally, Table V shows which is the technique that has performed better for each instance. In this case, the techniques that have used the NN as initialization function have outperformed the other alternatives in the 86.66% (13 out of 15) of the cases. Additionally, the supremacy of GA_{50} can also be seen in this table, having obtained the best solution in 80% of the instances.

IV. CONCLUSIONS AND FURTHER WORK

In this paper a study on the influence of using initialization functions in genetic algorithms applied to combinatorial optimization problems is performed. In this first stage of the research, the experimentation has been conducted with the well-known TSP. This experimentation has been carried out with three different heuristic functions. For each operator, the performance of four different GAs are compared. As final conclusion of this research it can be highlighted the

TABLE V. BEST HEURISTIC FUNCTION FOR EACH INSTANCE

Instance	Best heuristic	Instance	Best heuristic
Oliver30	NN (GA_{50})	KroC100	NN (GA_{50})
Eilon50	NN (GA_{10})	Eil101	NN (GA_{50})
Berlin52	NN (GA_{50})	Pr107	NN (GA_{50})
St70	NN (GA_{50})	Pr124	I1 (GA_{50})
Eilon75	NN (GA_{50})	Pr136	I1 (GA_{50})
Eil76	NN (GA_{50})	Pr144	NN (GA_{50})
KroA100	NN (GA_{100})	Pr152	NN (GA_{50})
KroB100	NN (GA_{10})		

efficiency of using heuristic initialization functions. Anyway, the excessive use of them could decrease the exploration capacity of the GA, trapping the population in local optimums quickly. Therefore, the key is to maintain a balance between individuals initialized by functions, and individuals generated randomly.

As future work, this experiment will be conducted with some other combinatorial optimization problems. In this way, it will be concluded whether the reasoning of this study are extendable to other problems.

REFERENCES

[1] L. Duan, M. K. Doğru, U. Özen, and J. C. Beck, "A negotiation framework for linked combinatorial optimization problems," *Autonomous Agents and Multi-Agent Systems*, vol. 25, no. 1, pp. 158–182, 2012.

[2] K. Smith-Miles and L. Lopes, "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research*, vol. 39, no. 5, pp. 875–889, 2012.

[3] E. Lawler, J. Lenstra, A. Kan, and D. Shmoys, *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley New York, 1985, vol. 3.

[4] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[5] E. G. Coffman and J. L. Bruno, *Computer and job-shop scheduling theory*. John Wiley & Sons, 1976.

[6] J. Lenstra and A. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[7] S. Kirkpatrick, C. Gellat, and M. Vecchi, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[8] F. Glover, "Tabu search, part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[9] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.

[10] K. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, Michigan, USA, 1975.

[11] E. Osaba, F. Diaz, and E. Onieva, "A novel meta-heuristic based on soccer concepts to solve routing problems," in *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion*. ACM, 2013, pp. 1743–1744.

[12] Z. Li, Z. Zhou, X. Sun, and D. Guo, "Comparative study of artificial bee colony algorithms with heuristic swap operators for traveling salesman problem," in *Intelligent Computing Theories and Technology*. Springer, 2013, pp. 224–233.

[13] E. Osaba, E. Onieva, R. Carballedo, F. Diaz, and A. Perallos, "An adaptive multi-crossover population algorithm for solving routing problems," in *Nature Inspired Cooperative Strategies for Optimization*. Springer, 2014, pp. 113–124.

[14] E. Osaba, R. Carballedo, F. Diaz, and A. Perallos, "Analysis of the suitability of using blind crossover operators in genetic algorithms for solving routing problems," in *8th International Symposium on Applied Computational Intelligence and Informatics*. IEEE, 2013, pp. 17–22.

[15] G. Mattos Ribeiro and G. Laporte, "An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem," *Computers & Operations Research*, vol. 39, no. 3, pp. 728–735, 2012.

[16] A. Bortfeldt, "A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints," *Computers & Operations Research*, vol. 39, no. 9, pp. 2248–2257, 2012.

[17] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[18] Z. H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," *International Journal of Biometrics & Bioinformatics (IJBB)*, vol. 3, no. 6, p. 96, 2010.

[19] M. Albayrak and N. Allahverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1313–1320, 2011.

[20] T. Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press Oxford, 1996, vol. 996.

[21] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs paralleles, reseaux et systems repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[22] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (abc) algorithm and applications," *Artificial Intelligence Review*, pp. 1–37, 2012.

[23] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.

[24] H. Nazif and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Applied Mathematical Modelling*, vol. 36, no. 5, pp. 2110–2117, 2012.

[25] J. C. Chen, C.-C. Wu, C.-W. Chen, and K.-H. Chen,

"Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm," *Expert Systems with Applications*, vol. 39, no. 11, pp. 10 016–10 021, 2012.

[26] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994.

[27] A. Kumar, N. Jani, P. Gupta, S. Saxena, S. Singh, S. Dhami, V. Singh, S. Kapoor, S. Singh, S. Chikara *et al.*, "An empirical study on crossover operator for degree constraint minimal spanning tree problem using genetic algorithm," *International Journal of Computational Intelligence Research*, vol. 8, no. 1, pp. 1–15, 2012.

[28] W. Banzhaf, F. D. Francone, and P. Nordin, "The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets," in *4th International Conference on Parallel Problem Solving from Nature*. Springer, 1996, pp. 300–309.

[29] E. S. Mresa and L. Bottaci, "Efficiency of mutation operators and selective mutation strategies: An empirical study," *Software Testing Verification and Reliability*, vol. 9, no. 4, pp. 205–232, 1999.

[30] J. Fernandez-Prieto, M. Gadeo-Martos, J. R. Velasco *et al.*, "Optimisation of control parameters for genetic algorithms to test computer networks under realistic traffic loads," *Applied Soft Computing*, vol. 11, no. 4, pp. 3744–3752, 2011.

[31] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[32] J. T. Alander, "On optimal population size of genetic algorithms," in *Proceedings on Computer Systems and Software Engineering*, 1992, pp. 65–70.

[33] R. L. Haupt, "Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors," in *International Symposium on Antennas and Propagation Society*, vol. 2. IEEE, 2000, pp. 1034–1037.

[34] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427–441, 2011.

[35] J. Le Ny, E. Feron, and E. Frazzoli, "On the dubins traveling salesman problem," *Automatic Control, IEEE Transactions on*, vol. 57, no. 1, pp. 265–270, 2012.

[36] B. Bontoux, C. Artigues, and D. Feillet, "A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 37, no. 11, pp. 1844–1852, 2010.

[37] M. Fischetti, J. J. S. Gonzalez, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, no. 3, pp. 378–394, 1997.

[38] E. Osaba and F. Diaz, "Comparison of a memetic algorithm and a tabu search algorithm for the traveling salesman problem," in *Federated Conference on Computer Science and Information Systems*. IEEE, 2012, pp. 131–136.

[39] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[40] E. Osaba, E. Onieva, F. Diaz, R. Carballedo, and A. Perallos, "Comments on albayrak, m., & allahverdy n.(2011). development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. expert systems with applications, 38 (3), 1313–1320: A proposal of good practice," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1530–1531, 2014.

[41] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 129–170, 1999.

[42] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs paralleles, reseaux et systems repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[43] M. Tomassini, "A survey of genetic algorithms," *Annual Reviews of Computational Physics*, vol. 3, no. 2, pp. 87–118, 1995.

[44] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.

[45] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the international joint conference on artificial intelligence*, vol. 1. Los Angeles, CA, USA, 1985, pp. 161–163.

[46] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Technical Journal*, vol. 44, no. 10, pp. 2245–2269, 1965.

[47] M. Englert, H. Röglin, and B. Vöcking, "Worst case and probabilistic analysis of the 2-opt algorithm for the tsp," *Algorithmica*, vol. 68, no. 1, pp. 190–264, 2014.

[48] E. Osaba, E. Onieva, R. Carballedo, F. Diaz, A. Perallos, and X. Zhang, "A multi-crossover and adaptive island based population algorithm for solving routing problems," *Journal of Zhejiang University SCIENCE C*, vol. 14, no. 11, pp. 815–821, 2013.

[49] D. P. Qu, H. Tu, and T. S. Fan, "Performance analysis of local optimization algorithms in traveling salesman problem," *Advanced Materials Research*, vol. 846, pp. 1364–1367, 2014.

[50] E. Osaba, F. Diaz, and E. Onieva, "Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts," *Applied Intelligence*, pp. 1–22, 2014.

[51] G. Reinelt, "Tspliba traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.