Analysis of the suitability of using blind crossover operators in genetic algorithms for solving routing problems

Eneko Osaba, Roberto Carballedo, Fernando Díaz and Asier Perallos
Deusto Institute of Technology.
University of Deusto, Av. Universidades, 24.
Bilbao, Spain.
Email: {e.osaba, roberto.carballedo, fernando.diaz, perallos}@deusto.es

***Abstract*— Genetic algorithms (GA) are one of the most successful techniques in solving combinatorial optimization problems. Its general character has enabled its application to different types of problems: vehicle routing, planning, scheduling, etc... This article shows that there is controversy in the basic structure of the algorithm steps when it is applied at routing problems. Specifically in this paper we show that the crossover (CX) offers no advantage in the optimization process. To solve such problems, the most important steps are mutation and selection of individuals. These two steps are what help to analyze the solution space exhaustively and give GA optimization capability. To prove our hypothesis we will analyze the results obtained by applying different blind crossover operators to solve multiple instances of the TSP (Travelling Salesman Problem).**

Keywords—Genetic algorithm, Crossover Operator, Optimization.

I. Introduction

Since its proposal in the '70s, genetic algorithms (GA) have become one of the most successful meta-heuristics for solving combinatorial optimization problems. Over the last decades GAs have been the focus of a large number of papers and books [1, 2] and it has been applied in a wide range of fields, for example, transport [3], software engineering [4] or industry [5].

A meta-heuristic is a technique designed to optimize a problem by iteratively trying to improve one or more solutions. Meta-heuristics do not use any information of the problem to get better results and they can explore very large spaces of solutions. This last point is what differentiates a meta-heuristic from a heuristic. A heuristic uses specific information of the problem to increase its capacity of optimization. It explores less the space of solutions and intensifies the search in areas considered most interesting.

Genetic algorithms are based on the genetic process of living organisms and in the law of the evolution of species, proposed by Darwin [6]. In the real world, over generations, populations evolve according to natural selection and survival of the strongest specimens. The GA was proposed in an attempt to imitate this natural process. The basic principles of this technique were proposed by Holland [7], even though its practical use for solving complex problems was shown by De Jong [8] and Goldberg [9]. The power of the GA is given by its robustness and its capacity of adaptability to a wide variety of problems of different areas.

The aim of this paper is to demonstrate the inefficiency of one of the central steps of the GA in the capacity of optimizing of the algorithm. This step is the crossover (CX), and to prove our hypothesis, we will perform an experiment in which we compare the results and times obtained by using the same GA but with different CX, with the results obtained by an evolutionary algorithm which performs no CX phase and focus its execution only in the process of mutation and selection of survivors.

The structure of this paper is as follows. In the following section we introduce shortly the way of operation of the GA. Next, we give a brief introduction of the problem that we choose to carry out our experiment, Travelling Salesman Problem (TSP) and we give some reasons for its election. Then, we will explain our hypothesis. Next we will describe the experiment and we will show the tests phase. We finish this paper with the conclusions of the study and further work.

II. Genetic Algorithms

The GA is aimed to solve combinatorial optimization problems, and as we explain in the introduction, it tries to find a good solution basing on the evolution and the genetic of natural species. The GA is widely known by the scientific community, for this reason we will not extend on its introduction. With the intention of completing information, we will only explain schematically the conventional implementation of a GA. For more detailed information about the GA, lot of sources can be found in the literature [2, 9, 10].

Algorithm 1 execution process of a GA

```
1: Build the initial population
2: While (Termination Criterion not reached){
3:   Select parents from the population;
4:   Crossover with the parents selected;
5:   Mutation process;
6:   Selection of survivors;
7: }
8: Return the best individual of the population;
```

III. THE TRAVELING SALESMAN PROBLEM, TSP

The traveling salesman problem, or TSP [11], is one of the most famous and widely studied problems throughout history in operations research and computer science. The TSP can be defined on a complete directed graph $G = (V, A)$ where $V = \{v_0, v_1, \ldots, v_n\}$ is the set of vertices which represents the clients of the system, and $A = \{(v_i, v_j): v_i, v_j \in V, i \neq j\}$ is the set of arcs which represents the interconnection between clients. Each arc has a distance cost d_{ij} associated, which is at a known distance matrix C. The TSP objective is to find a route that visits each and every customer once and that minimizes the total distance traveled.

A. Formulation

The problem can be formulated as follows [12]:

$$Minimize \sum_{(i,j)\in A} d_{ij}\, x_{ij} \quad (1)$$

$$Subject\ to$$

$$\sum_{j\in\Delta+(i)} x_{ij} = 1, \qquad \forall i \in V \quad (2)$$

$$\sum_{i\in\Delta-(j)} x_{ij} = 1, \qquad \forall j \in V \quad (3)$$

$$\sum_{i\in S, j\in\Delta+(i)\setminus S} x_{ij} \geq 1, \qquad \forall S \subset V \quad (4)$$

$$x_{ij} \in \{0,1\}, \qquad \forall \{i,j\} \in E \quad (5)$$

Being x_{ij} a binary variable which is 1 if the arc (i,j) is used in the solution. Furthermore, V is the set of the customers of the system and d_{ij} is the distance between the nodes i and j. The objective function, (1), is the sum of all the arcs in the solution used, or what is the same, is the total distance of the route. This objective function has to be minimized. Constraints (2) and (3) indicate that each node has to be visited and abandoned only once, while Section (4) guarantees the absence of sub-tours and indicates that any subset of nodes S has to be abandoned at least 1 time. This restriction is vital, because it avoids the presence of cycles.

B. Reasons of choosing the TSP

The TSP has been extensively studied throughout history and has been treated with a multitude of different techniques. It has a great scientific interest and today is used in a large number of studies [13-15]. In this article we use this problem not to demonstrate that the GA is a good alternative to solve the TSP, something already proven. We do it to demonstrate our hypothesis, which we will explain in the next section. Therefore, we have used this problem because it is well known, and it is simple to implement and understand. In addition, it is easily replicable, so that any reader of this article can prove this same experiment, either to check their results or to try to verify the same hypothesis but with other CX. Another advantage of using the TSP as a problem to be treated is that there are a lot of CX for solving this problem, which can be easily found in a variety of studies, for example [16], and more recently [17].

IV. HYPOTHESIS

This article aims to demonstrate the hypothesis that we propose:

> *"Crossover phase of the genetic algorithms is not efficient for the search process and the capacity of optimization of the technique when it is applied to routing problems using path encoding".*

For it, we should first explain a factor that may be controversial if we not mention it.

To test the ability of optimization of a meta-heuristic to solve any combinatorial optimization problem, it is appropriate to use neutral operators throughout the implementation of it. In other words, as we said in the introduction, operators that use characteristics of the problem and optimize by themselves have to be avoided.

As an example of this we can mention the initialization process of the GA. The most appropriate way to prove the quality of optimization of a meta-heuristic is to use a random initialization process, instead of using initialization functions, such as those proposed by Solomon in 1987 [18]. If we use any of these initialization functions, the individuals will pass through an optimization process before the execution of the GA generations. Therefore, we may not know exactly what the capacity of optimization of a meta-heuristic is when we obtain the final results that it gives. In this case, we could say that we are implementing a heuristic, because we use specific information of the problem.

In this way, extrapolating this fact to the CX phase, the most suitable form to demonstrate that this phase is inefficient is using neutral crossover functions, which only take care of generating individuals that meet the constraints of the problem and not optimize by themselves.

This is why the operators like the Very Greedy Crossover (VGX) [19] obtain better results than other neutral operators. The VGX is an operator for the TSP that uses the distances between cities to generate the children resulting from the crossing. Is logical to think that using this operator, the GA will get good results for the TSP, as the VGX makes by itself a small optimization on the resulting individuals. If we use this kind of functions, we have to say that we are implementing heuristics, instead of meta-heuristics. Thus, we will use 100% neutral functions in CX, initialization and mutation phases in our experiments.

Another reason why the use of the CX functions is not beneficial is the complexity of creating combinations of individuals that satisfy the constraints of the problem. This occurs especially in problems with strict restrictions, like the Vehicle Routing Problem with Time Windows [20]. There are many studies that after creating the children using the CX, use a process to "fix" the resulting infeasible solutions [21]. These processes of arrangement can be beneficial, but they increase the complexity and time of execution of the

technique. Moreover, they undermine the philosophy of the CX functions, since the children resulting are not "natural" combinations of their parents.

V. Experiments and analysis of results

For the experiments, a conventional GA has been used, introduced earlier in this article. The initial population is generated 100% randomly. The parents and survival selection criterias are 100% elitist for all experiments (meaning that best individuals, based on their fitness value, have the priority to be selected), with the exception of the experiment 7, which uses a 50% elitist-random survivor criterion. Solutions are encoded using the Path Representation [22]. As a mutation function, the widely known 2-opt has been used [23]. This function has been widely used in many studies along history [24, 25]. As explained above, there are a large number of CX operators for the TSP, in the work of Larrañaga et al [22] a large number of them are collected. In this study we use four CX functions, three of them have been widely used since its creation, for that reason have been selected for this study. These functions are Order Crossover (OX) [10], Modified Order Crossover (MOX) [26] and Order Based Crossover (OBX) [27]. Finally, the last operator used is a particular case of the traditional crossover, in which the cut point is made always in the middle of the path. We have called it Half Crossover, and it could be explained as follows.

Half Crossover (HX): As we said, in this crossover, a cut is made in the central position of the parents. For example:

P = (1 2 3 4 | 5 6 7 8)

M = (2 4 6 8 | 7 5 3 1)

The order of cities that are to the left of the cutting point remains in the corresponding order in the offspring. The remaining cities are added in the same position that found in the other parent. Thus, the result of the process would be as follows:

P = (1 2 3 4 | 6 8 7 5)

M = (2 4 6 8 | 1 3 5 7)

Continuing with the characteristics of GA, for all tests we used a population of 48 individuals, and the number of the generation of each execution is proportional to the neighborhood of the mutation function.

A. Results

The following tables show the results obtained by the GA in the different experiments made. Those tests were performed on an Intel Core i5 – 2410 laptop, with 2.30 GHz and a RAM of 4 GB. For each run we display the total average, the worst and the best of the results obtained and the standard deviation. We also show the average of the execution time, in seconds. The number of executions for each instance is 50. Instances were obtained from the TSP Benchmark TSPLIB [28]. The name of each instance has a number that displays the number of nodes it has.

Each table shows the results obtained by a specific algorithm, and each of the tests differs in the type of CX used. The first four experiments show different GA with different CX, all with a crossover and mutation probability of 100%, which means that all individuals perform a crossover and a mutation every generation. In the experiment 5 a GA without any CX was used, utilizing only the mutation function, also with a probability of 100%. This last table will serve to make an accurate comparison with which we will draw the conclusions to be discussed in the next section. The different tables explain its features in its own title.

TABLE I.
EXPERIMENT 1. GA WITH THE FUNCTION OX AND 2-OPT MUTATION

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	425.3	9.86	420	448	0.55
Eilon50	442.9	7.43	430	454	2.52
Eil51	448.3	8.69	437	463	2.76
Berlin52	7945.2	262.71	7596	8476	2.86
St70	709.4	13.53	687	729	8.5
Eilon75	578.2	5.37	571	590	9.5
Eil76	581.5	12.39	565	598	15.35
KroA100	22265.7	581.74	21545	23095	26.43
KroB100	23602.9	413.89	22884	24373	26.26
KroC100	21850.1	465.5	21280	22727	26.93
Eil101	683.7	10.45	665	698	30.25
Pr107	46676.5	1406.5	45083	49167	37.56
Pr124	60852.6	1288.63	59302	62877	50.12

TABLE II.
EXPERIMENT 2. GA WITH THE FUNCTION MOX AND 2-OPT MUTATION

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	423.5	5.97	420	437	0.63
Eilon50	445.9	7.94	433	456	2.28
Eil51	450.3	6.62	440	461	2.45
Berlin52	8062.4	168.4	7782	8327	2.6
St70	721.4	17.73	705	757	6.85
Eilon75	582.0	5.85	572	593	9.34
Eil76	585.5	14.98	568	623	9.57
KroA100	22445.3	687.87	21555	23580	24.86
KroB100	23599.5	878.01	22604	25017	24.56
KroC100	22146.4	341.3	21362	22518	25.75
Eil101	692.9	11.18	681	722	26.54
Pr107	46076.5	1043.5	44960	48511	34.97
Pr124	61399.0	1263.7	59397	63362	42.76

TABLE III.
EXPERIMENT 3. GA WITH THE FUNCTION OBX AND 2-OPT MUTATION

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	426.1	8.57	420	446	0.49
Eilon50	445.4	5.87	438	455	1.64
Eil51	448.7	7.12	437	458	1.70
Berlin52	8027.6	267.1	7542	8435	1.76
St70	718.5	23.98	690	765	4.62
Eilon75	579.6	16.74	562	610	6.12
Eil76	577.5	8.73	557	585	6.37
KroA100	22690.0	473.76	21790	23291	15.42
KroB100	23548.0	577.06	22833	24521	15.67
KroC100	22710.0	875.78	21659	23933	16.54
Eil101	683.0	9.68	553	695	17.33
Pr107	46682.2	1105.01	45486	48688	23.19
Pr124	61282.6	1832.66	59657	65115	36.00

TABLE IV.
EXPERIMENT 4. GA WITH THE FUNCTION HX AND 2-OPT MUTATION

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	427.2	8.99	420	449	0.4
Eilon50	451.3	7.42	443	467	1.74
Eil51	451.1	8.57	442	465	1.52
Berlin52	7988.4	251.72	7542	8399	1.59
St70	714.2	12.03	598	739	5.02
Eilon75	575.9	7.62	567	595	6.85
Eil76	584.6	14.59	565	611	6.22
KroA100	22195.8	381.69	21675	23060	18.94
KroB100	23366.6	533.15	22675	24194	20.92
KroC100	21995.8	593.22	20977	22941	19.94
Eil101	696.0	12.35	679	723	21.12
Pr107	46866.9	964.11	45945	49173	28.9
Pr124	61002.2	1361.5	59868	63560	48.55

TABLE V.
EXPERIMENT 5. GA WITHOUT CX, BUT WITH 2-OPT MUTATION

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	423.1	2.81	420	428	0.1
Eilon50	448.0	8.00	435	460	0.3
Eil51	449.1	7.49	438	462	0.29
Berlin52	8023.5	357.31	7542	8412	0.31
St70	712.6	13.98	700	742	0.65
Eilon75	583.4	13.49	563	605	0.79
Eil76	577.4	8.37	565	591	0.84
KroA100	21856.9	309.93	21400	22432	1.89
KroB100	23194.6	304.66	22722	23717	1.91
KroC100	21680.2	447.95	20933	22319	1.88
Eil101	689.8	14.83	670	714	1.92
Pr107	45584.0	802.83	44678	46840	2.12
Pr124	61040.8	1389.97	59199	63886	3.64

To facilitate the comparison between different experiments, the next table shows the averages of each of the tests. The best mean of each instance is bolded, while the other values have a percentage of the deviation regarding this best average:

TABLE VI.
COMPARISON BETWEEN THE AVERAGES OF THE EXPERIMENTS

Instance	Exp. 1	Exp. 2	Exp. 3	Exp. 4	Exp. 5
Oliver30	425.3 (0.51%)	423.5 (0.09%)	426.1 (0.70%)	427.2 (0.96%)	**423.1**
Eilon50	**442.9**	445.9 (0.67%)	445.4 (0.56%)	451.3 (1.89%)	448.0 (1.15%)
Eil51	**448.3**	450.3 (0.44%)	448.7 (0.08%)	451.1 (0.62%)	449.1 (0.17%)
Berlin52	**7945.2**	8062.4 (1.47%)	8027.6 (1.03%)	7988.4 (0.54%)	8023.5 (0.98%)
St70	**709.4**	721.4 (1.69%)	718.5 (1.28%)	714.2 (0.67%)	712.6 (0.45%)
Eilon75	578.2 (0.39%)	582.0 (1.05%)	579.6 (0.64%)	**575.9**	583.4 (1.30%)
Eil76	581.5 (0.71%)	585.5 (1.40%)	577.5 (0.01%)	584.6 (1.24%)	**577.4**
KroA100	22265.7 (1.87%)	22445.3 (2.69%)	22690.0 (3.81%)	22195.8 (1.55%)	**21856.9**
KroB100	23602.9 (1.76%)	23599.5 (1.74%)	23548.0 (1.52%)	23366.6 (0.74%)	**23194.6**
KroC100	21850.1 (0.78%)	22146.4 (2.15%)	22710.0 (4.74%)	21995.8 (1.45%)	**21680.2**
Eil101	683.7 (0.10%)	692.9 (1.44%)	**683.0**	696.0 (1.90%)	689.8 (0.99%)
Pr107	46676.5 (2.39%)	46076.5 (1.08%)	46682.2 (2.40%)	46866.9 (2.81%)	**45584.0**
Pr124	**60852.6**	61399.0 (0.89%)	61282.6 (0.70%)	61002.2 (0.24%)	61040.8 (0.31%)

These tests have been performed comparing an evolutionary algorithm without CX and with a probability of mutation of 100%, with 4 different versions of GA in which the probability of crossover and mutation is 100%. Now, to extend the tests, we will show the results from experiment 6, in which we used a GA with crossover probability of 90%, mutation probability of 5%, and a 100% elitist survival selection criterion. For this experiment the OX and 2-opt functions are used, the first for the crossovers and the second for the mutations. After this, we show another experiment with the same GA, but instead using a 100% elitist survival criterion, a 50% elitist-random criterion is used.

TABLE VII.
EXP. 6. GA WITH THE FUNCTION OX 90% AND 2-OPT MUTATION 5%.

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	428.4	8.03	420	443	0.62
Eilon50	456.2	8.89	442	471	1.68
Eil51	459.6	19.25	443	500	1.52
Berlin52	7912.8	227.81	7604	8355	1.75
St70	738.2	22.00	718	788	4.86
Eilon75	590.4	12.83	569	612	6.58
Eil76	606.1	18.77	582	639	6.86
KroA100	22366.1	512.87	21671	23186	16.86
KroB100	23252.5	487.89	22592	23992	18.15
KroC100	22077.9	851.97	21177	23498	17.54
Eil101	726.9	26.27	702	785	18.85
Pr107	47220.4	1225.6	45059	49101	25.54
Pr124	59414.4	6941.5	40178	64487	40.73

TABLE VIII.
EXP. 7. GA WITH FUNCTION OX WITH A 90% OF PROBABILITY, 2-OPT MUTATION WITH 5% AND 50% ELITIST-RANDOM SURVIVOR FUNCTION

Instance	Avg.	S. dev.	Best r.	Worst r.	Time
Oliver30	427.0	12.94	420	453	0.62
Eilon50	457.0	11.58	439	476	1.83
Eil51	451.7	14.79	437	484	1.75
Berlin52	7843.6	229.64	7542	8135	1.58
St70	720.6	17.00	690	753	5.12
Eilon75	586.6	15.88	572	620	6.47
Eil76	588.2	11.14	573	605	7.12
KroA100	22316.7	614.86	21469	23379	17.54
KroB100	23026.5	500.14	22306	23983	17.21
KroC100	21591.2	724.96	21000	23320	17.38
Eil101	708.4	12.19	684	727	19.15
Pr107	46481	1332.4	44526	59625	24.57
Pr124	60428.3	911.87	59597	62387	39.91

As we have done before, below we show a table comparing the results of the last experiments with the experiment in which only the mutation was used. As in the table 6, the best average is bolded and the other values show the deviation.

TABLE IX.
COMPARISON BETWEEN AVERAGES OF THE EXPERIMENTS 6, 7 AND 5.

Instance	Exp. 6	Exp. 7	Exp. 5
Oliver30	428.4 (1.25%)	427.0 (0.92%)	**423.1**

Eilon50	456.2 (1.83%)	457.0 (2.00%)	**448.0**
Eil51	459.6 (2.33%)	451.7 (0.60%)	**449.1**
Berlin52	7912.8 (0.88%)	**7843.6**	8023.5 (2.29%)
St70	738.2 (3.60%)	720.6 (1.12%)	**712.6**
Eilon75	590.4 (1.19%)	586.6 (0.54%)	**583.4**
Eil76	606.1 (4.97%)	588.2 (1.87%)	**577.4**
KroA100	22366.1 (2.32%)	22316.7 (2.10%)	**21856.9**
KroB100	23252.5 (0.98%)	**23026.5**	23194.6 (0.73%)
KroC100	22077.9 (2.25%)	**21591.2**	21680.2 (0.41%)
Eil101	726.9 (5.37%)	708.4 (2.69%)	**689.8**
Pr107	47220.4 (3.58%)	46481 (1.96%)	**45584.0**
Pr124	**59414.4**	60428.3 (1.70%)	61040.8 (2.73%)

B. Analysis of the results and conclusions

Viewing the results presented in these tables the conclusions that can be drawn are clear. In the first four tables the results obtained by the GA with the different functions of CX are shown, being all results very similar to each other. This way, we cannot be able to conclude roundly which one gets better results and which is the better. Regarding the execution times, HX, MOX and OX obtain very similar times, while OBX requires less effort to its executions. Introducing the experiment 5 we can see that using a GA without crossover the results are as good as those obtained using any of the CX previously used, obtaining in some cases better results than the other 4 algorithms. Despite this, if we look at the percentages of the deviations of the means, the vast majority is less than 2%, and rarely exceeds the 2.5%, so that the differences between them are minimal. This means that no algorithm offers any improvement that makes it better than the rest. In addition, regarding the run times, these are much lower in the experiment 5 than in the other experiments. This difference is not very noticeable in problems with few nodes, as Oliver30. But when the nodes increases, the differences become larger, reaching an increase between 800% and 1000% respect the times of the experiment 5, in instances of 100 or more nodes. This happens because the routing problems are classified as NP-Hard problems [29].

After the first 5 experiments, two new experiments are shown, which use a version of GA with more conventional mutation and crossover probabilities, and in which different functions are used for the survivors selection. The results of these new experiments lead to the same conclusions as the above, there is no perceptible improvement in the results obtained by these GA and the results obtained by the evolutionary algorithm of experiment 5.

Looking at the results, we can draw the following conclusions, which support our hypothesis:

- **Conclusion 1:** The use of CX functions do not give any improvement to the process of optimizing of a GA, being the other functions (mutation and selection of survivors) which provide the ability of optimization to the GA.
- **Conclusion 2:** Using this kind of functions increases considerably the execution time. Furthermore, as the number of nodes grows, the time rises exponentially. This last point can be seen by comparing the time in instances as KroA100, KroB100 or Pr124. This is especially important for real-time applications, where the execution time becomes decisive.

These two statements can lead to a new reasoning that supports our theory

- **Conclusion 3:** The crossover phase increases the complexity of the GA algorithm without providing any visible improvement.

Conclusions 2 and 3 can be obtained easily, since the more steps a meta-heuristic has, more time needs for execution and more complex is to design and develop. On the other hand, conclusion 1 can be based on several arguments. The main purpose of the CX functions is to obtain new individuals making combinations of the characteristics of the individuals of the population. As explained before, to test the ability of optimizing of meta-heuristic neutral functions must be used, otherwise, the optimization capability may lie in these functions. Thus, it is logical to think that the CX can hardly meet its target, since it is very improbable that the resulting offspring from neutral crosses improve their parents. It is for this reason that the main utility of CX might be to increase the capability of the search process of the algorithm. The use of this kind of functions or mechanisms can be very beneficial for a meta-heuristic, as can be seen in other techniques such as simulated annealing [30], which uses a cooling process to allow this type of jumps inside the space of solutions, or the tabu search [31], with the mechanisms of memory at medium and long term. Despite this, the use of this type of functions must not be excessive, since the jumps in the solution space are beneficial only in determined times, for example when a local optima is reached.

In GA, mutations can handle this type of sudden jumps in the solution space. In addition the mutation can also make small jumps on the solution space, which are very positive for the optimization process. This fact can free the GA of using CX functions and it can reduce its running time, achieving similar results, as we can see in the tables shown

With all this, we can say that our hypothesis has been verified, providing evidence to certify it, and arguments that support it. Finally, as a last conclusion we could enter the next reasoning which is supported by the results shown in the experiments.

- **Conclusion 4:** The most efficient way to implement an evolutionary algorithm to solve routing problems is basing it only on the functions of mutation and selection of survivors.

Despite this, the idea of combining characteristic of more than one individual of the population could be useful. If these mechanisms were used in very specific moments, they could be beneficial to the process of exploration of the solution space. For this, the way of using these functions would need to radically be changed, and the philosophy of the meta-heuristic should change regarding the GA. As future work, we will design and implement a new population-based meta-heuristic focused on local searches, which uses specific CX functions only in exceptional cases, when we certain that it benefits the process of crawl of the space of solutions.

REFERENCES

[1] Harvey, I. The Microbial Genetic Algorithm. Advances in Artificial Life. Darwin Meets von Neuman, 5778: 126-133, 2011.

[2] Affenzeller, A., Winkler, S., Wagner, S. and Beham, A. Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. Chapman & Hall/CRC, 2009.

[3] Moon, I., Lee, J.H., and Seong, J. Vehicle routing problem with time windows considering overtime and outsourcing vehicles. Expert System with Applications, 39: 13202-13213, 2012.

[4] Martinez-Torres, M.R. A genetic search of patterns of behavior in OSS communities. Expert System with Applications, 39: 13182-13192, 2012.

[5] Chen, J.C., Wu, C.-C., Chen, C.-W. and Chen, K.-H. Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm. Expert System with Applications, 39: 10016-10021, 2012.

[6] Darwing, C. On the Origin of Species by Means of Natural Selection. Murray-London, 1859.

[7] Holland, J.H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press-Cambridge, 1975.

[8] De Jong, K. A. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. diss., University of Michigan, Michigan, USA, 1975.

[9] Goldberg, D. Genetic algorithms in search, optimization and machine learning. Addison Wesley Publishing Company Inc.-New York, 1989.

[10] Davis, L. Applying Adaptive Algorithms to Epistatic Domains. In: Proc. Of the International Joint Conference on Artificial Intelligence, 162-164, 1985.

[11] Lawler, E.L., Lenstra, J.K., Rinnooy-Kan, A.H.G. and Shmoys, D.B. The Traveling Salesman Problem: A guided tour of combinatorial optimization. Wiley-New York, 1985.

[12] Dantzing, G., Fulkerson, D. and Johnson, S. Solution of a large scale traveling salesman problem. Operations research, 2: 393-410, 1954.

[13] Liegooghe, A., Humeau, J., Mesmoudi, S., Journdan, L., and Talbi, E.G. On dominance-based multiobjetive local search: desing, implementation and experimental analysis on scheduling and traveling salesman problems. Journal of Heuristics, 18: 317-352, 2011.

[14] Osaba, E., and Diaz, F. Comparison of a memetic algorithm and a tabu search algorithm for the traveling salesman problem. In: Federated Conference on Computer Science and Information Systems, FedCSIS. 131-136. 2012.

[15] Casazza, M., Ceselli, A. and Nunkesser, M. Efficient algorithms for the double traveling salesman problem with multiple stacks. Computers & operations Research, 39: 1044-1053, 2012.

[16] Oliver, I., Smith, D. and Holland, J.R. A study of permutation crossover operators on the traveling salesman problem, in: Proc. 2nd Int. Conf. on Genetic Algorithms. 224-230.Lawrence Erlbaum-New Jersey, 1987.

[17] Ray, S. Bandyopadhyay, S. and Pal, S. New Genetic operators for solving TSP: Application to microarray gene ordening. Springer-Berlin, 2005.

[18] Solomon M.M. Algorithms for the vehicle routing and scheduling problems with time windows. IFORMS Operations Research, 35: 254-265, 1987.

[19] Julstrom, B.A. Very Greedy Crossover in a Genetic Algorithm for the TSP. In: Proc of the 1995 ACM symposium on Applied Computing, 324-328, 1995.

[20] Bräysy, O. and Gendreau, M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39: 104-118, 2005.

[21] Jung, S. and Moon, B.-R. A hybrid genetic algorithm for the vehicle routing problem with time windows. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, 1309-1316. 2002.

[22] Larrañaga, P., Kuijpers, C.M.H., Murga, R.H., Innza, I. and Dizdarevic, S. Genetic Algorithms for the Traveling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review, 13: 129-170, 1999.

[23] Lin, S. Computer Solution to the traveling salesman problem. Bell System Technical Journal, 44: 2245-2269, 1965.

[24] Tarantilis, C.D., and Kiranoudis C.T. A flexible adaptive memory-based algorithm for the real-life transportation operations: Two case studies form diary and construction sector. European Journal of Operational Research, 179, 806-822, 2007.

[25] Bianchessi, N., and Righini, G. Heuristic algorithm for the vehicle routing problem with simultaneous pick-up and delivery. Computers & Operations Research, 34, 578-594, 2007.

[26] Ray, S. Bandyopadhyay, S. and Pal, S. New Operators of Genetic Algorithm for Traveling Salesman Problem. In: Proc. Of the 17th International Conference on Pattern Recognition, 497-500, 2004.

[27] Syswerda, G. Schedule Optimization Using Genetic Algorithms. In Davis, L (ed.) Handbook of Genetic Algorithms, 332-349. Van Nostrand Reinhold-New York, 1991.

[28] Reynelt, G. TSPLIB – A traveling salesman problem library. ORSA Journal on Computing, 3: 376-384, 1991.

[29] Lenstra, J.K. and Rinooy-Kan, A.H.G. Complexity of the vehicle routing and scheduling problems. INFORMS Networks, 11: 221-227. 1981.

[30] Van Laarhoven, P.J.M. and Aarts, E.H.L. Simulated Annealing: Theory and Applications. Reidel-Dordrecht, 1987.

[31] Glover, F. and Laguna, M. Tabu Search. Kluwer Academic Publishers-Boston, 1997.