Adv Syst Sci Appl 2020; **02**:32–43
Published online at https://ijassa.ipu.ru.

Dynamic Adaptation of Genetic Algorithm for Solving Routing Problems on Large Scale Systems

Viktor Zakharov[1], Alexander Mugayskikh[1]*

[1]*Saint Petersburg State University, Saint Petersburg, Russia*

Abstract: This paper is devoted to the implementation of the dynamic adaptation procedure for the genetic algorithm used for solving large-scale travelling salesman problem. This procedure serves to obtain more profitable solutions by a fixed operating time. In order to evaluate effectiveness of new approach computational experiments were performed on well-known problem instances from TSPLib library. As a result, generated solutions reduce the length of the routing plans in considered problem instances compare to classical genetic heuristics. By that, we show how to use the property of time inconsistency of heuristics to get better solutions. New criteria for estimating the efficiency of heuristics algorithms called experimental level of time consistency is introduced.

Keywords: time consistency, genetic algorithm, vehicle routing problem

1. INTRODUCTION

One of the most challenging goals in the field of transportation logistics is cost optimization, which allows to increase company's benefit for the same amount of resources used. For transportation companies, reduced costs can be achieved by constructing effective route plans for the vehicles. A significant effect of cost reduction can be shown on large transport networks. For this reason, researches pay much attention to models and methods which generate less expensive solutions in the routing problems. Probably, the most important objective of mathematical modelling in transportation logistics is the vehicle routing problem. A classical example of a vehicle routing problem is the travelling salesman problem, which will be examined in details.

This article is motivated by the problem of time inconsistency (or dynamic instability) of heuristic algorithms solving vehicle routing problems and developing methods to avoid this: to increase the level of dynamic stability and efficiency of the algorithms with experiments hold on a genetic algorithm. The article is structured as follows. After introduction, it includes seven sections. The first section is devoted to the description of general model of the travelling salesman problem (TSP) and contains the review of methods solving it. The second section provides the basic concepts and terms of the genetic algorithm used in the text. Section 3 includes a description of the basic scheme of the genetic algorithm and one of the most effective tools for constructing a solution — the Greffenstett's crossover. The fourth section discusses the problem of dynamic instability of heuristics in solving routing problems and a new criterion is suggested — the experimental level of time inconsistency. In section 5 we introduce the description of dynamic adaptation procedure which can improve basic solutions of heuristic algorithms. Section 6 presents the results of numerical experiments to evaluate

*Corresponding author: alexander.mugaiskih@gmail.com the level of time inconsistency of the genetic algorithm and a comparative analysis of the results obtained by the genetic heuristics and its dynamic adapdation procedure. Comparative analysis was hold on a set of problem instances from the TSPLib library, widely known among specialists. In the seventh section we place brief conclusions and plans for further research.

2. TRAVELLING SALESMAN PROBLEM: FORMULATION AND METHODS

2.1. Problem formulation

The travelling salesman problem consists in finding the shortest trip between a set of vertices, visiting each vertice exactly once, starting and finishing at a specified vertice.

The TSP can be represented as a graph $G = (V, E)$, where set of vertices $V = \{v_1, v_2, \dots, v_n\}$ stands for cities from the problem instance and set of edges $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ represents paths between the cities. The weight of an edge (v_i, v_j) is the distance or travel cost between vertices v_i and v_j.

Travelling salesman problem can be defined as an problem of finding a Hamiltonian cycle in graph G with the least weight. Let $I = \{1, \dots, n\}$ be the set of indices of vertices from the problem instance. The problem objective function f is total length of the route that includes all vertices from the problem instance. Suppose x_{ij} is a binary variable which is equal to one if route passes the arc (i, j), and zero otherwise. Length of the route between two vertices i and j is defined as Euclidean distance c_{ij}. Then the problem can be defined in the form of linear programming proposed by [9, 27, 29]:

$$f = \sum_{i \in I} \sum_{j \neq i, j \in J} c_{ij} z_{ij} \to min,$$

subject to:

$$0 \leq z_{ij} \leq 1, z_{ij} \in \mathbb{Z}$$

$$\sum_{j \in I, j \neq i} z_{ij} = 1, \forall i \in I, \tag{2.1}$$

$$\sum_{i \in I, i \neq j} z_{ij} = 1, \forall j \in I, \tag{2.2}$$

$$\sum_{i \in S, j \in S} z_{ij} = |S| - 1 \quad (S \subset V, |S| > 1) \tag{2.3}$$

Equalities (2.1) – (2.2) enforce that every vertice is visited exactly once. The last constraints guarantee the absence of two or more disjoined parts in the final route.

The travelling salesman problem takes a central role in combinatorial optimization. It has a wide range of practical application in various fields: transport logistics [1], robotics [6, 8], job shop scheduling. Among its applications one can mention constructing routing plans for a fleet of vehicles of logistic company; performing video surveillance by wireless robots [33]; robot-assisted sensor network deployment and data collection [37]; generating assembly sequences in an autonomous multi-robot coordinated furniture system [26]; planning routes in controlled airspace [12]. Despite the seeming simplicity of the mathematical model of the travelling salesman problem, it still receives close attention from specialists while considering the problem in new formulations and under various restrictions and constraints.

2.2. Algorithms and methods

All of approaches solving travelling salesman problem can be assigned either to exact or heuristics algorithms. Brute-force search, branch and bound algorithm, Branch and cut methods and dynamic programming are among the exact algorithms.

The branch-and-bound method and the branch-and-cut method represent the modification of exhaustive search. The main idea is to form the rooted tree, check the bounding criterion bounding and discard the current branch if it is not possible to produce a better solution than the best one found so far by the algorithm. Some variants of this approach can solve the travelling salesman problem with several hundred vertices, however, they require high computation power of the computers used.

The idea of dynamic programming, proposed by R. Bellman [3], M. Held, and R. Karp [19], can be also applied to the TSP problem and consists in a many-step decision-making process, at each step of which it is necessary to determine the Bellman function and find the best route for the left vertices. The number of operations required to calculate the final route increases exponentially with of size of problem instance.

Due to the NP-complexity of the TSP problem, exact methods cannot always be effectively applied to large-scale problems. For this reason, heuristic methods are used that generate solutions that are close to optimal, but in an acceptable time compared to exact algorithms. An experimental analysis of heuristic algorithms for solving the TSP problem and its subclasses was carried out in [17].

Heuristic algorithms can be divided into 2 classes according to the method of generating the final route: constructive and iterative heuristics. Tour construction heuristics obtain only one solution without its further improvement [24]. For example: nearest neighbour algorithm [23], greedy algorithm [18], insertion heuristics [14] and Christofides approximation algorithm presented in [7]. The algorithms above sequentially build a feasible solution by adding vertices to the final route until a complete route is formed. The part of the solution built to the current moment will remain unchanged until the end of the algorithm. The length of generated solutions is about 10–15% away from the optimal solution.

More useful for getting shorter routes are tour improvement heuristics. There methods begin to work with an already prepared route generated by one of the constructive methods, iteratively improving the solution at each step. There are several ways to do this: 2-opt swap and 3-opt swap, or perform as a part of local search algorithms and the Lin-Kernigan heuristics [20, 21].

Metaheuristics as a higher-level procedure also refer to tour improvement heuristics [4, 13]. These are quite general iterative procedures using randomization and self-learning elements, intensification and diversification of the search, adaptive control mechanisms, as well as constructive heuristics and local search methods. Metaheuristics work either with single candidate or with whole population of feasible solutions. Single solution approaches include variable neighborhood search, tabu search, simulated annealing [15] and others. Population-based approaches are genetic algorithms, ant colony optimization [10], greedy randomized adaptive search procedure and some others [36].

3. TERMINOLOGY AND DEFINITIONS

Further in the article, the following terms and definitions are used.

Heuristic algorithm is a problem-solving method, which solutions may not necessarily be optimal, but about which it is known that it gives a fairly good solution in most cases.

Genetic algorithm is a heuristic algorithm for solving optimization problems by randomly selecting, combining and varying the desired parameters using mechanisms similar to natural selection occurred in nature, generating different solutions at each launch due to the randomization mechanisms used in it.

Individual is a feasible solution to a problem instance in a genetic algorithm.

Population is the final set of individuals in a problem instance, which the genetic algorithm will work with.

Chromosome is a set of genes that describe an individual. In the travelling salesman problem, genes in the chromosome correspond to the vertices of the test problem. Chromosome contains the order in which the vertices are passed through in this solution.

Gene is an atomic element of a chromosome.

Crossover is the main stage of the genetic algorithm, in which two individuals of the same population take part. By combining and inheriting the characteristics of both parent individuals (parents), new individuals (children) are generated, and the quality of new individual is improved as well as the quality of whole offspring population.

Mutation is one of the stages of the genetic algorithm, in which one individual from the population takes part. Mutation is carried out with the aim of restoring the individuals dropped out of the population and forming genes that were not present in the original population.

Selection is the final stage of the genetic algorithm aims to form a new population of individuals by selecting the most adapted individuals for their inclusion in the further process of evolution.

4. GENETIC ALGORITHM WITH GREFFENSTTE'S CROSSOVER

The genetic algorithm is a metaheuristic algorithm, which is based on operations of natural selection that occurs in nature. It was first introduced in 1975 by John Holland as a subcategory of evolutionary algorithms [22]. Without detailing the description of all stages of the genetic algorithm, we briefly give a pseudocode of the classical genetic algorithm, which consists of the following steps:

```
1. Initialization of stating population, $s$ individuals.
2. While stop criterion is not met, do:
3. Several crossover operators (with 2 two random individuals)
4. Several mutation operators (random individual)
5. Evaluation of fitness function and selection
```

Applied to the travelling salesman problem, each individual in population stands for one feasible solution of a problem instance and represents the order in which the vertices are visited. The fittest individual in the last population is accepted as the final solution.

The genetic algorithm is a genetal heuristic procedure that can be implemented in a software package for solving different flavors of routing problems, such as TSPTW (Travelling salesman problem with time windows), IRP (Inventory routing problem), VRP (Vehicle routing problem), MDVRP (Multi-depot vehicle routing problem), PDP (Pickup and delivery problem) and others. In addition, genetic heuristics is capable of solving both the original problem and its sub-problem, which is necessary condition for applying dynamic adaptation procedures of this algorithm.

As it is known, the main operator of the genetic algorithm, which will be discussed in details in Section 5, is crossover. The crossover methods proposed in the literature are valid for the travelling salesman problem and give generally acceptable results, but they do not use information about the distance between the vertices of the TSP problem instances. John Greffenstett solved this problem by proposing in his work [16] a new type of crossover that uses information about the distance between vertices.

As applied to TSP intances with a symmetric distance matrix of 70 and 100 vertices of the TSPLib test library [32], the use of the Greffenstett's heuristic at the stage of crossover helped to get shorter routes compared to partially mapped crossover, cycle crossover, edge recombination crossover, two-point crossover, uniform order-based crossover, shuffle crossover and sub-tour exchange crossover. For more information on these types of crossovers, see [25]. For a problem instance with 100 vertices, Greffenstett's crossover showed the best result among other kinds of crossovers and differs from optimal solution by 11.9% [25]

In work [31] authors also compared 8 types of crossover to solve the VRP problem, generalization of the TSP problem on the number of vehicles (2 and more). The following crossovers were described: order crossover, partially mapped crossover, edge recombination crossover, cycle crossover, alternating edges crossover, Greffenstett's crossover and its two modifications [31]. The best results among the mentioned algorithms on the VRP test problems of the Christofides standard library [11] were shown by the Greffenstett's heuristic crossover and its two modifications.

Thus, the proposed kind of crossover outperforms its alternatives in generating shorter routes, and it was implemented in the genetic algorithm for solving problem instances for further research. The results are presented in the following sections of the article.

The following scheme presents the classic Greffenstett crossover operation:

```
1. Randomly chosen vertice $v$ is placed in the first gene of the children chromosome
2. **While** complete route is not formed, **do:**
3. Compare the length of two edges connected with $v$, choose the shortest among them and
   denote the corresponding vertice as $v_{min}$
4. **If** $v_{min}$ is already in children's chromosome, **then** $v_{min}$ will be the randomly chosen
   vertice in the set of non-used in the current route
5. Place vertice $v_{min}$ in the next gene of children's individual
   $v = v_{min}$.
```

5. TIME INCONSISTENCY OF HEURISTIC ALGORITHMS

In work [5], the authors note that the process of finding the optimal solution in genetic algorithms is guided exclusively by the obtained values of the objective function at the previous points in the solution space. And assumptions about such properties of the objective function such as convexity, differentiability, and Bellman's optimality property optimality are not used.

Formulated in [2] principle of optimality claims that at any period of time restriction of multi-period optimal routing plan found by exact algorithm for any remaining time horizon appears to be optimal in current sub-problem. It's worth noting, restriction of solution obtained by tour improvement heuristic algorithm could be not optimal in a current sub-problem for at least one of the periods. Constructive heuristics almost always generate time consistent routes, however, they do not provide a high level of efficiency compared to iterative ones.

Using the principles of dynamic stability described in [30], we introduce the following notation.

Let us consider set of problem instances P for the travelling salesman problem. Suppose that for each instance $p \in P$ we can obtain a set of different solutions $S(p)$ generated by heuristics. Each solution $s(p)$ corresponds to an order in which vertices of the problem instance are visited in a route. We consider the order of the vertices of one route and divide it into T parts so that the number of vertices that were visited in the first t parts of the route is calculated by the formula $n(t, s(p)) = \lfloor n_0 t/T \rfloor$, where n_0 is the size of the problem instance p, and T — parameter defined before the experiment. Thus, each part of the route (except the last) contains the same number of vertices. By the period t, where $t = 0, 1, \dots, T-1$, we mean the time interval in the route that corresponds to the part t of the original solution $s(p)$.

Define $s^+(t,p)$ as remaining consequence of nodes after period t and $s^-(t,p)$ stands for such part of the route that includes nodes visited during periods $\tau = 0, 1, \dots, t$. Thus each solution can be represented as $s(p) = s^-(t,p) \cup s^+(t,p)$.

For $t = 1, \dots, T-1$ we consider the sub-problem $p(s^-(t,p))$. This sub-problem differs from initial problem instance. Firstly, nodes already visited before period t are excluded for visit in next periods. Secondly, new depot is located at the last node of part $s^-(t,p)$ that belongs to the route $s(p)$. Then we obtain solution $s(p(s^-(t,p)))$ for the sub-problem $p(s^-(t,p))$ using the same heuristics.

Definition 5.1:
Solution $s(p)$ (routing plan), obtained by heuristic algorithm H, is time consistent according to algorithm H, if for each $t = 1, \dots, T-1$ and each $s(p(s^-(t,p)))$ generated by H, the following inequality holds:

$$f(s^+(t,p)) \leqslant f(s(p(s^-(t,p)))), \tag{5.4}$$

where f is distance value for corresponding route.

Definition 5.2:
Solution $s(p)$ (routing plan), obtained by heuristic algorithm H, is time inconsistent according to algorithm H, if exists t' and $s(p(s^-(t',p)))$ generated by H, and the following inequality holds:

$$f(s^+(t,p)) > f(s(p(s^-(t',p)))). \tag{5.5}$$

The property of time consistency of the solution in the TSP shows that moving according to the initial route plan, no shorter solution will be found than the current one in the sub-problem generated by the method described above. And vice versa, time inconsistency of the solution means that at least in one of the periods a routing plan can be generated that will more profitable than the current one.

We introduce the procedure for calculating the level of time consistency of the heuristic algorithm H for a certain class of vehicle routing problems.

1. Consider set P of problem instances of certain class in routing problems.
2. Generate set N of various solutions for every $p \in P$ by heuristic algorithm H. Obtained solutions (routing plans) will be different from each other due to the randomization used in heuristic operators: crossover, mutation, selection.
3. Perform M runs for each solution $s(p) \in S(p)$, where each run includes checking whether solution is time consistent or not. Parameter M aims to average the obtained results of calculation, assume $M = 5$. Starting with $t = 1$, formulate sub-problem $p(s^-(t,p))$ and find its solution by algorithm H. Check initial solution to be time consistent after first period. If inequality (5.4) holds, move to the second period. If not, mark the period at which property of time consistency is violated and start the next run.

Let $b(s,t)$ be the number of runs, in which time consistency is violated for the solution $s(p)$ for period t. If s is optimal, then regarding Bellman's principle of optimality we would have the following quality:

$$\sum_{t=1}^{T-1} b(s,t) = 0. \tag{5.6}$$

Definition 5.3:
Experimental level of time consistency $conH$ for heuristics H to be the value calculated as follows

$$conH = 1 - \frac{1}{M|P|} \sum_{p \in P} \frac{1}{|N|} \sum_{s(p) \in N} \sum_{t}^{T-1} b(s(p),t). \tag{5.7}$$

The experimental level of time consistency can vary for problem instances but one can note that $0 \leq conH \leq 1$. High value of experimental level of time consistency for heuristics forms expectation that considered heuristics generates solutions which tend to stay optimal during the realization of its initial routing plan.

6. DYNAMIC ADAPTATION OF GENETIC ALGORITHM

The concept of dynamic stability has been widely studied in recent years for problems of game theory. Using the main idea of the algorithm proposed in [40], we suggest dynamic adaptation procedure for the class of the travelling salesman problem and genetic algorithm solving it.

At the initial stage, we generate a set of N different solutions of the problem instance by the genetic algorithm. From the set N we choose the best solution, with the minimum value of the objective function. For it, we will obtain M experiments on checking if property of time consistency holds. If this property is violated, starting from the period t, i.e. after the period t, the better solution will be found in the current subproblem $p(s^-(t,p))$. To reduce the total length of the routing plan through all vertices, the current routing plan should be changed according to the new solution obtained in the considered subproblem. The general scheme of dynamic adaptation is as follows [28].

```
1. Obtain set $N$ of different solutions generated by heuristics $H$ for $p \in P$.
2. Define $s_1(p) = \arg\min_{s(p) \in N} f(s(p))$.
3. From $t$ = 1 **to** $T - 1$ **do**
4. Formulate sub-problem $p(s_t^-(t,p))$, obtain set $N$ of solution.
5. Define $s_{t+1}^*(p) = \arg\min_{s(p(s_t^-(t,p))) \in N} f(s(p(s_t^-(t,p))))$.
6. Check if the property of time consistency is valid for $s_t(p)$.
7. **If** $f(s_t^+(t,p)) > f(s_{t+1}^*(p))$ , **then** routing plan should be changed to $s_{t+1}(p) = s_t^-(t,p) \cup s_{t+1}^*(p)$.
```

7. COMPUTATIONAL RESULTS

7.1. Experimental level of time consistency of the genetic algorithm with Greffenstett's crossover for solving the TSP

To conduct the experiment, five test instances from the standard TSPLib library were considered and a genetic algorithm with a Greffenstett's heuristic crossover to solve them was applied. The following set of parameters was used in the computational experiment: $|P| = 5, M = 5, T = 5, |N| = 20$. For each of the test problems, 20 different solutions were generated. The number of runs for one solution $s(p)$ equaled 5. The original route was divided into five periods. The period number t was marked, after which the solution $s(p)$ lost the time consistency property. These values correspond to the columns $b(s,t)$ in Table 7.1.

Table 7.1. The experimental level of time consistency of the GA for the TSP

Problem instance	Number of runs	$b(s,t)$				Number of time consistent runs
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	
att48	100	31	16	26	8	19
eil51	100	64	14	11	3	8
berlin52	100	19	27	4	4	46
st70	100	16	22	4	1	57
eil101	100	15	28	3	2	52
Sum	500	145	107	48	18	182

The average level of Experimental level of time consistency of the genetic algorithm for solving the TSP equals $0,364$.

$$conGA = 0{,}364.$$

This value is quite low: only a third of the solutions generated at the initial stage keep the optimality property in the process of their implementation. This means that there are other routes that can be obtained dynamically using the same heuristic algorithm, the values of the objective function of which will be less than in current solutions.

We calculate the level of time consistency for solutions obtained using the dynamic adaptation procedure. We will carry out the experiment according to the same scheme and with the parameters determined earlier. The experimental results are shown in Table 7.2.

Table 7.2. The experimental level of time consistency of the dynamic adaptation for the GA for the TSP

Problem instance	Number of runs	$b(s,t)$				Number of time consist. runs
		$t=1$	$t=2$	$t=3$	$t=4$	
att48	100	24	5	11	1	59
eil51	100	47	11	7	5	30
berlin52	100	2	12	5	3	78
st70	100	1	7	5	5	82
eil101	100	1	6	4	4	85
Sum	500	75	41	32	18	334

The average value of the experimental level of time consistency of a dynamically adapted genetic algorithm for solving the TSP problem is 0.668, which is almost two times higher than for the genetic algorithm before adaptation.

We consider the dependence of the number of time-consistent solutions obtained during the experiment in the first t periods on period t. The total number of runs performed is $M|N||P|$. Function (8) is introduced which states for the number of time-consistent solutions after the completion of each period.

$$c(s,t) = M|N||P| - \sum_{k=0}^{t} b(s,k). \tag{7.8}$$

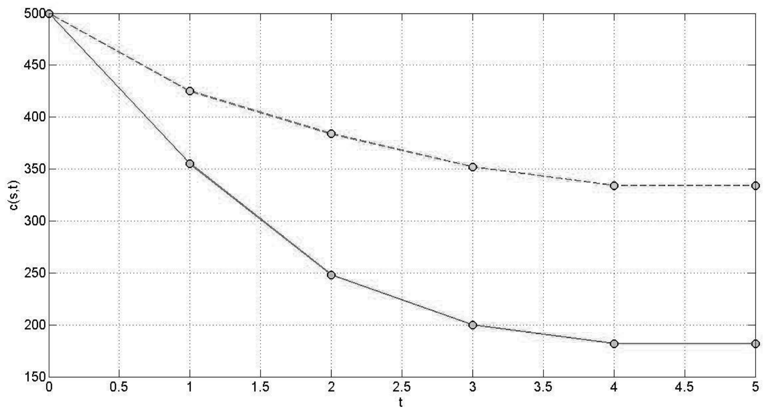

Fig. 7.1. Number of time consistent solution after period t for the genetic algorithm and its dynamic adaptation

In Figure 7.1, the continuous line corresponds to the values of the function $c(s,t)$ during calculation the experimental level of the time consistency of the genetic algorithm with the Greffenstett's heuristic crossover, and the dashed line corresponds to the values of this function for the dynamic adaptation of this heuristics.

7.2. Comparison of route plans obtained by genetic heuristics and its dynamic adaptation procedute

Using the example of the TSP problem and the genetic algorithm, it was shown that the dynamic adaptation procedure allows to obtain solutions with a higher level of time consistency.

We will check how the length of the route plans changes with this procedure. Table 7.3 presents a comparison of the lengths of routes (solutions) generated by the genetic algorithm and its dynamic adaptation. The average values of the objective function are given for 500 runs of algorithms. The stopping criterion for each start of the genetic algorithm with the Greffenstett's heuristic crossover was reaching 100 generations of population. The dynamic adaptation procedure was carried out according to the scheme given in algorithm 3, with the following set of parameters: $T = 5, |N| = 20$.

We introduce the following notation: the value $l_{GA}(p)$ equals the length of the route obtained by the genetic algorithm for the test problem p, $l_{DAGA}(p)$ is the length of the route obtained by using a dynamically adapted genetic algorithm. The last two columns of table 3 show the values of the objective function corresponding to the effective start. An effective run of the dynamic adaptation procedure for the problem instance p is a run for which the value of $l_{GA}(p) - l_{DAGA}(p)$ is the maximum in the experiment.

Table 7.3. Comparison of length of routing plans generated by genetics and its dynamic adaptation

Problem instance	Mean length value		Effective run	
	Genetic heuristics	Dynamtic adaptation	Genetic heuristics	Dynamtic adaptation
att48	12270,72	11723,79	11657	10928
eil51	493,72	464,64	493	448
berlin52	8836,31	8257,36	8570	7893
st70	803,25	758,61	765	708
eil101	779,30	720,338	725	665

Table 3 shows that for each of the problem instances considered, the average length of the generated solutions by the dynamic adaptation algorithm is less than the classical genetic algorithm with the Greffenstett's heuristic crossover. The percent of improvement we calculate by the equation below:

$$k = \frac{l_{GA}(p) - l_{DAGA}(p)}{l_{GA}(p)} 100\%. \tag{7.9}$$

The average percentage improvement is $6,01\%$. The maximum improvement value was obtained for the problem instance eil51 and amounts to $9,12\%$.

7.3. Time analysis for applying dynamic adaptation procedure

The procedure of dynamic adaption of a genetic algorithm requires additional runs on each period and leads to increasing the time to get one routing plan. We will carry out the following experiment. Let t_{dyn} be the average duration (in seconds) of the dynamic adaptation algorithm to generate a route. Let us limit the runtime of classical genetic heuristics to t_{dyn}. For each problem instance, we will make 100 launches of the genetic algorithm and its dynamic adaptation and compare the results of the two methods with a fixed duration of their work equal to t_{dyn}.

As can be clearly seen from Table 7.4, with a fixed operating time, the method of dynamic adaptation of genetic heuristics is able to generate better solutions compared to the genetic algorithm with the Greffenstett's heuristic crossover. This remark should be taken into account in large-scale problem instances, when the generation time of one route can take

Table 7.4. Comparison of length of routing plans obtained by fixed runtime t_{dyn}

Problem instance	Mean length value		t_{dyn}
	Genetic alrorithm	Dynamic adaptation	
att48	12273,51	11723,79	120,1
eil51	492,44	464,64	165,5
berlin52	8820, 25	8257,36	206,35
st70	807,37	758,61	1300,4
eil101	786,54	720,338	3403,7

tens of minutes. The experiment also showed, in the case of a fixed runtime for generating the routing plan, it is more profitable to use the dynamic adaptation procedure instead of the classical heuristic, distributing the time between the corresponding periods of the algorithm.

8. CONCLUSION

In this article, a genetic algorithm with a Greffenstett's heuristic crossover solving the TSP problem was described and implemented in Java programming language. The level of time consistency of the solutions generated by this heuristic is estimated. The calculations for a set of problem problems from the TSPLib library demonstrated that the genetic algorithm with the Greffenstett's heuristic crossover generates routes with a low level of time consistency.

To generate solutions with a higher level of time solvency, a dynamic adaptation procedure for a genetic algorithm with a Greffenstett's heuristic crossover is proposed. As a result, the level of time consistency of the new solutions increased by two times. At the same time, the average value of the route length decreased by 6.01%. Thus, the practical application of the proposed dynamic approach allows to generate shorter routes.

It was also shown that with a fixed restriction on the operating time, the procedure for dynamically adapting the genetic algorithm with the Greffenstett's crossover leads to obtain shorter routes.

It is worth noting that the proposed method of dynamic adaptation can be applied to other heuristic algorithms for solving the TSP problem. The usage of the this method to solve vehicle routing problems such as TSPTW, IRP, MDVRP, PDP is possible if the heuristic algorithm is able to solve both the problem itself and its subproblems [34, 35].

REFERENCES

1. Anbuudayasankar S. P., Ganesh K., Mohapatra S. (2014). Survey of Methodologies for TSP and VRP, *Models for Practical Routing Problems in Logistics*, Springer, 11–42.
2. Bellman R. (1957). *Dynamic Programming*, Princeton: Princeton University Press, 1957. – 392 p.
3. Bellman R. (1962). Dynamic programming treatment of the travelling salesman problem, *Journal of the ACM*, 9, 61–63.
4. Blum C., Roli A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Comput. Surv.*, 35(3), 268–308.
5. Borisovskiy P.A., Eremeev A.V. (2004). Comparison of some evolutionary algorithms, *Automation and Remote Control*, 3, 3–9, [in Russian].
6. Chiu K.-M., Liu J.-S. (2011). Robot routing using clustering-based parallel genetic algorithm with migration, *Proceedings of Merging Fields Of Computational Intelligence And Sensor Technology*, 9, 42–49.
7. Christofides N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem, *Technical Report 388, Graduate School of Industrial Administration*, Carnegie

Mellon University, 11 p.

8. Comarela, G., Goncalves, K., Pappa, G.L., Almeida, J., Almeida, V. (2011). Robot routing in sparse wireless sensor networks with continuous ant colony optimization, *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation GECCO'11*, New York, NY, USA, ACM, 599–606.
9. Dantzig G.B. (1963). *Linear programming and extensions*, Princeton: Princeton Univ. Press, 219 p.
10. Dorigo M., Gambardella L.M. (1997). Ant colonies for the travelling salesman problem, *Biosystems*, 43(2), 73–81.
11. Dorronsoro B. The VRP Web. *Languages and Computation Sciences Department*, University of Malaga, http://www.bernabe.dorronsoro.es/vrp/
12. Furini F., Persiani C.A., Toth P. (2016). The Time Dependent Travelling Salesman Planning Problem in Controlled Airspace, *Transportation Research Part B: Methodological*, 90, 38–55.
13. Gendreau M., Potvin J.-Y. (2010). *Handbook of Metaheuristics*, Springer Publishing Company, 649 p.
14. Gendreau M., YeHertzo A., Laporte G., Stan M. (1998). A Generalized Insertion Heuristic for the Travelling Salesman Problem with Time Windows, *Oper. Res.*, 46(3), 330–335.
15. Granville V., Krivanek M., Rasson J.-P. (1994). Simulated annealing: A proof of convergence, *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(6), 652–656.
16. Grefenstette J. J., Gopal R., Rosmaita B. J., Van Gucht D. (1985). Genetic algorithms for the traveling salesman problem, *Proceedings of the 1st International Conference on Genetic Algorithms*, Hillsdale: Lawrence Erlbaum Associates, 160–168.
17. Gutin G., Punnen A.P. (2002). *The traveling salesman problem and its variations*, Kluwer Academi, 837 p.
18. Gutin G., Yeo A., Zverovich A. (2002). Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp, *Discrete Applied Mathematics*, 117(1–3), 81–86.
19. Held M., Karp R.M. (1961). A dynamic programming approach to sequencing problems, *Proceedings of the 1961 16th ACM National Meeting, ACM*, 201–204.
20. Helsgaun K. (2000). An effective implementation of the lin-kernighan traveling salesman heuristic, *European Journal of Operational Research*, 126, 106–130.
21. Helsgaun K. (2009). General k-opt submoves for the lin-kernighan tsp heuristic, *Math. Program. Comput.*, 1(2–3), 119–163.
22. Holland J. H. (1975). *Adaptation in natural and artificial systems*, Ann Arbor: The University of Michigan Press, 183 p.
23. Hurkens Cor A. J., Woeginger G.J. (2004). On the nearest neighbor rule for the traveling salesman problem *Oper. Res. Lett.*, 32(1), 1–4.
24. Johnson D.S., McGeoch L.A. (2001). *Experimental analysis of heuristics for the stsp. In Local Search in Combinatorial Optimization*, Wiley and Sons, 80 p.
25. Khan I.H. (2015). Assessing Different Crossover Operators for Travelling Salesman Problem, *I.J. Intelligent Systems and Applications*, 1, 19–25.
26. Knepper, R.A., Layton, T., Romanishin, J., Rus, D. (2013). Ikeabot: An autonomous multi-robot coordinated furniture assembly system, *Robotics and Automation*, 855–862.
27. Miller C.E., Tucker A. W., and Zemlin R.A. (1960). Integer programming formulation of traveling salesman problems *J. ACM*, 7(4), 326–329.
28. Mugayskikh A.V. (2015). Dynamic adaptation of genetic algorithm for travelling salesman problem, *Control Processes and Stability*, 2, 665–670, [in Russian].
29. Papadimitriou C.H. and Steiglitz K. (1982). *Combinatorial Optimization: Algorithms and Complexity*, Upper Saddle River: Prentice-Hall, Inc. 496 p.
30. Petrosyan L.A., Zenkevich N.A. (2009). Principles of dynamic stability, *Large-Scale Systems Control*, 3, 100–120, [in Russian].

31. Puljic K., Manger C.R. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem, *Mathematical Communications*, 18, 359–375.
32. Reinelt G. *Travelling Salesman Problem Library*, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
33. Sivasoundari A., Kalaimani S. (2013). Wireless surveillance robot with motion detection and live video transmission, *International Journal of Emerging Science and Engineering*, 1, 147–165.
34. Shirokikh V.A., Zakharov V.V. (2015). Dynamic Adaptive Large Neighbourhood Search for Inventory Routing Problem, *Advances in Intelligent Systems and Computing*, Springer, 359, 231–241.
35. Shirokikh V.A., Zakharov V.V. (2017). Heuristic evaluation of the characteristic function in the Cooperative Inventory Routing Game, *Journal on Vehicle Routing Algorithms*, Springer, 1–14.
36. Talbi E.-G. (2013). *Metaheuristics for Bi-level Optimization*, Springer Publishing Company, Incorporated, 288 p.
37. Wang, Y., Wu, C.H. (2007). Robot-assisted sensor network deployment and data collection, *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 467–472.
38. Zachariasen M, Dam M. (1996). Tabu Search on the Geometric Traveling Salesman Problem, *Proceedings from Metaheuristics International Conference*, Colorado, 1996, 571–587.
39. Zakharov V.V., Dementieva M. (2004). Multistage cooperative games and problem of time consistency, *International Game Theory Review*, 6, 157–170.
40. Zakharov V.V., Shchegryaev A.N. (2014). Multi-period cooperative vehicle routing games, *Contributions to Game Theory and Management*, 7(2), 349–359.
41. Zakharov V.V., Shchegryaev A.N. (2015). Stable Cooperation in Dynamic Vehicle Routing Problems, *Automation and Remote Control*, Springer, 76(5), 935–943.