Ant colony optimization with parameter update using a genetic algorithm for travelling salesman problem

Ekaterina A. Blagoveshchenskaya[a], Ilya I. Mikulik[a], Lutz H. Strüngmann[b]
[a]*Petersburg State Transport University, 9 Moskovsky pr, Saint Petersburg, 190031, Russian Federation*
[b]*Mannheim University of Applied Sciences, 10 Paul-Wittsack st, 68163 Mannheim, Germany*

Abstract
It is widely known that the ant colony algorithm is sensitive to changes in parameters. We present a novel kind of the algorithm which uses a genetic algorithm for solving this problem. Travelling salesman problem, known as NP-complete problem, is one of the most popular and important tasks in combinatorial optimization. This is a typical task for testing combinatorial optimization algorithms. In this paper, we present a comparative analysis of the algorithm with a simple ant colony optimization (SACO) on the example of this problem.
Keywords
Ant colony optimization, genetic algorithm, travelling salesman problem

1. Introduction

Ant colony optimization is one of the popular methods for combinatorial problem-solving. It has practical applications for logistics tasks [1][2], dispatching tasks [3], cloud service [4], route planning [5], multiple sequence alignment [6], and others. The algorithm belongs to a class of so-called bio-inspired algorithms, whose behaviour is inspired by examples from nature[1].

A common disadvantage of the ant colony class of methods is sensitivity to parameters that establish a balance between expanding the search space and exploiting the solutions found. Optimal parameters can be highly problem-specific and dependent on the required accuracy of the solution. Therefore, some works are devoted to the dynamic adaptation of the parameters of the ant algorithm [7]. We decided to use this method and use a genetic algorithm to update the parameters.

The proposed algorithm is hybrid since it is based on two algorithms. In contrast to some other hybrid algorithms [6] [8], the proposed algorithm is not a sequential application of several algorithms. On the other side, it is not a fundamentally new algorithm. The idea is to apply genetic optimization of the parameters of the main, ant colony optimization.

The study was conducted on the TSP problem. It was chosen because there exists an amount of literature to benchmark our results against. TCP is to find the cheapest way of visiting all of the cities and returning to starting point. The benchmark berlin52 was used as an example.

In our work, we considered simple ant colony optimization (SACO) as the simplest implementation of the ant colony optimization, for two reasons. The first is to evaluate how well the genetic algorithm finds the acceptable parameters of the ant algorithm. The second is to determine how much the found parameters improve the algorithm's performance. Using SACO eliminates the possibility of improving the quality of the algorithm due to some property of a specific implementation, a variant of the algorithm. It is possible that this approach of finding parameters with another method of the ant colony optimization will show a better result.

Models and Methods for Researching Information Systems in Transport, Dec 11-12, St. Petersburg ,Russia
EMAIL: kblag2002@yahoo.com (E.A. Blagoveshchenskaya); mikulik.ilia@gmail.com (I.I. Mikulik); l.struengmann@hs-mannheim.de(L.H. Strüngmann);
ORCID: 0000-0002-2425-5556 (E.A. Blagoveshchenskaya); 0000-0002-0542-7914 (I.I. Mikulik); 0000-0002-1689-3611 (L.H. Strüngmann)

2. Travelling salesman problem

The Travelling Salesman Problem (TSP) is a representative of class of problems known as combinatorial optimization problems. The task can be formulated as follows. The graph of cities is given. A salesman must visit all the cities only once. He must build a tour such that the length of the common path is the shortest among all possible paths. Path is a set of edges. Each edge from the graph has a weight or length. The length of the path is the sum of weights all edges from this path.

In terms of graph theory, the problem can be formulated as finding the shortest Hamiltonian cycle.

The problem is NP-hard and transcomputational, so finding an exact solution to the problem is not always acceptable due to time and computational resource constraints. In this case, approximate algorithms that give suboptimal solutions are often used in practice. Algorithms differ both in the resources consumed (time, memory) and in the accuracy of the approximation.

The traveling salesman problem occupies a special place in combinatorial optimization and operations research. Historically, it was one of tasks that served as an examples for the development of these areas. The simplicity of the formulation, the finiteness of the set of acceptable solutions, the visibility and, at the same time, the enormous cost of a complete search still motivate mathematicians to develop new numerical methods.

The travelling salesman problem is a routing problem and has practical application. The main goal in these tasks is to build routes for vehicles that serve a certain set of customers. At the same time, choosing an unsuccessful route entails additional costs for transporting the goods.

The above reasons explain our choice of task. Since the problem is a typical one, there are a sufficient number of examples of graphs with optimal solutions already found in the public domain. This makes it easy to compare the results of the work done with the existing ones. One such example is the berlin52 set on which the study was conducted.

3. Ant colony optimization and genetic algorithm

Ant colony optimization (ACO) and genetic algorithm (GA) are population-based search algorithms by maintaining a population of structures as main elements in their design.

The algorithms belong to metaheuristic class. They make few assumptions about the optimization problem being solved and so they can be used for different problems solving. These algorithms do not guarantee the accuracy of the solution and can be applied in problems where one can be satisfied with suboptimal solutions.

3.1. Ant colony optimization

The ant colony algorithm, based on Swarm intelligence, is a heuristic algorithm. It simulates the behaviour of an ant colony when searching for food. This model is used for solving combinatorial optimization problems.

In the natural world, ants randomly search for food. When they find food, they return to the colony, leaving traces with pheromones. Pheromones serve as a signal to other ants. If other ants find a path with pheromones, they will most likely follow it. If the ants get to the food source, they will also start leaving pheromones on the way back. Over time, the pheromone begins to evaporate, so the trodden paths become less interesting for ants. The shorter the path, the less pheromone has time to evaporate during the passage of the ant from the food source to the anthill. Passing along such a path becomes fast, and the pheromone density remains high.

Evaporation plays an important role in finding global minima: if pheromones evaporate at a low rate, the path found by the algorithm will represent a local minimum. Thus, when one ant finds the optimal (or suboptimal) path from the food source to the anthill, other ants are more likely to choose the same path, and positive feedback eventually leads all ants to the same, shortest path.

The formulation of the model already hints at the sensitivity of the algorithm to parameters: the evaporation coefficient, the amount of pheromone deposited, the sensitivity of the ant to pheromones.

Ant colony optimization is a broad class of algorithms. There are a big number of variations of this algorithm. They differ in the edge selection rule, the pheromone deposition rule, and the evaporation rule. In some implementations, there are elite ants. In such implementations, only the elite ants who have built the best paths have the right to put a pheromone on the arc. Algorithms can also

differ in the number of ants: it can be static or dynamic. However, most of the proposed implementations are more or less parameter-dependent. Of interest are algorithms whose parameters are dynamic, they change during the operation of the algorithm. The proposed algorithm is also dynamic in terms of changing parameters.

3.2. Genetic algorithm

A genetic algorithm (GA) is a heuristic algorithm that models natural selection. Like ACO, GA is often used for combinatorial optimization problems. GA uses principles and terminology borrowed from genetics. In GA, each individual represents a potential solution to some problem and is encoded in a special way (in the simplest case, a binary number). Set of individuals, which are potential solutions, form the population.

The search for the optimal solution to a problem in GA is performed in the process of population evolution. It is a sequential transformation of one finite set of solutions into another using the genetic operators of selection, crossover, and mutation.

The idea of the selection operator is that individuals with better fitness function values have a better chance of surviving and reproducing. The fitness function is defined over the genetic representation and it measures the quality of the represented solution. There are many ways to implement the selection operator.

The crossing operator allows to get new individuals by crossing old ones. It combines the information of two parents to generate new offspring. Just like the selection operator, the crossover operator can be implemented in various ways. The crossover operator has its own implementation depending on the type of data that the algorithm works with.

The mutation operator allows to get fundamentally new individuals. Usually, the mutation operator changes a small part of the genome (part of the parent's data) with a low probability.

The advantage of the algorithm is that the evolutionary idea can be applied in different problems and in different ways.

Another advantage that GA can be hybridized with other optimization methods for improving their performance such as ACO, simulated annealing and other. Simulating annealing is a metaheuristic which approximates global optimization. It is based on physical temperature annealing, which is well studied [11]. Simulating annealing can also be applied for parameter optimization.

The main advantages of hybridized GA are better solution quality, better efficiency, guarantee of feasible solutions, and optimized control parameters.

4. The proposed GA-ACO algorithm

The proposed algorithm inspired by a simple ant colony optimization (SACO). GA is used for parameters search.

Every ant k has parameters α_k, β_k,Q_k.

These parameters are shared in SACO, but in our implementation they are different for each ant. For these parameters, the optimal values are searched using the GA algorithm.

α_k is sensitivity. It determines how sensitive the ant is to the pheromone. This number is an exponent degree. It can be not integer.

β_k is heuristic sensitivity. It determines how sensitive the ant is to the heuristic information. This number is also an exponent degree. It can be not integer. If the number is zero, the heuristic information is ignored.

Q_k is pheromone intensity. It determines how much pheromone the ant will put on the passed edges. This is an integer. This number strongly depends on the choice of the pheromone deposition rule.

Another static common parameters are n, ρ, τ_0.

n is an ants count. This is an important parameter. It affects not only the accuracy of the solution, but also the execution time of the algorithm. It cannot be changeable by GA because our implementation involves working only with the parameters that may differ for each ant.

τ_0 is small pheromone value. It is upper bound of pheromone distribution during initialization. It also cannot be changeable because it only plays a role once.

ρ is a rate of evaporation. This is an important parameter. However, this parameter applies to the entire graph as a whole. The parameter cannot be bound to a specific ant. Therefore, despite the extreme importance of this parameter in finding the optimal solution, we cannot use a genetic algorithm to change it.

Each ant chooses an edge probabilistically, according to the rule

$$p_{ij}^{k}(t) = \frac{\tau_{ij}^{\alpha_k}(t)\eta_{ij}^{\beta_k}(t)}{\sum \tau_{iu}^{\alpha_k}(t)\eta_{iu}^{\beta_k}(t)} \quad (1)$$

where $p_{ij}^{k}(t)$ is a probability of choosing edge iJ by ant k at t iteration.

Each ant puts a pheromone according to the rule

$$\Delta\tau_{iJ} = \frac{Q_k}{f(x_k)} \quad (2)$$

where $\Delta\tau_{iJ}$ is a pheromone value; $f(x_k)$ is a length of the path x which was found by ant k.

Algorithm can be formed by the next steps:

Begin

Determine the parameters: n, τ_0, ρ.

For each edge ij:

assign a random amount of pheromone $\tau_{ij} < \tau_0$.

While stop criteria is not reached:

For each ant $k = 1..n$:

$x_k(t) = \emptyset$.

While the full path is not built:

Set the next vertex j according to the rule (1).

$x_k(t) = x_k(t) \cup (i,j)$.

For each edge (i,j):

calculate the evaporation and the new pheromone concentration by the rule (2).

Create new set of ant's parameters according the selection rule.

Use the crossover operator for parameters.

Use the mutation operator for parameters.

Set new parameters α_k, β_k,Q_k to ants.

End

The main idea of the proposed algorithm is the use of genetic operators. Firstly, it is needed to define the fitness function, because the fitness function is always problem dependent. It was decided to define the fitness function as the length of the path traversed by the ant. Note that we use a genetic algorithm to optimize the parameters, not the found path. However, the path length is a good indicator for evaluating how well the ant colony algorithm works for the given parameters.

In this paper we use selection rule which defines a new set of parameters probabilistically:

$$P(M_k) = \frac{f(x_k)}{\sum f(x_k)}, \quad (3)$$

where $P(M_k)$ is a probability to choose ant's k parameters. Actually, any appropriate rule can be applied here. We use roulette wheel selection. Crossover operator implements bit crossing of selected parameters. Mutation operator randomly changes a bit of each parameter.

This genetic part allows to change parameters and find the best of them in real time.

5. Results

We conducted an experiment on berlin52 dataset. According to the article [11], the optimal parameters for the berlin52 dataset are $\alpha = 0.7$, $\beta = 4$, $\rho = 0.5$. Also, To compare the results of the SACO with non-optimal parameters we give another set of parameters $\alpha = 0.2$, $\beta = 1$, $q = 15$, $\rho = 0.9$. SACO with this parameter set gives worse result then presented before. We used the presented algorithm with these ones as initial.

As result, the parameters in the last iterations of our method have become closer to optimal. An example of the path constructed by the proposed algorithm is shown in the Figure 1.

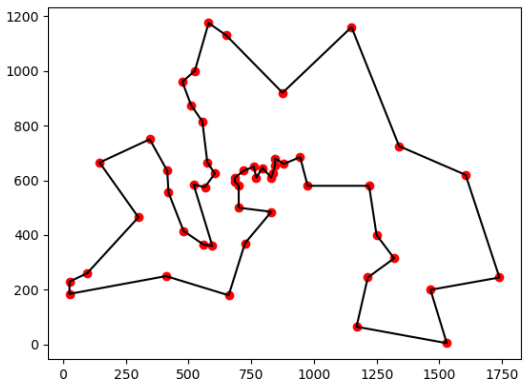

Figure 1: solution for berlin52 dataset

A comparison of the results of ACO with optimal parameters, ACO with non-optimal parameters, and the proposed ACO+GA algorithm is presented in table 1.

Table 1
Comparison

Algorithm	Best solution	Average solution
ACO with not optimal parameters	11178.291	11883.161
ACO with optimal parameters	7548.993	7816.862
ACO+GA	7548.993	7998.069

The maximum, minimum, and average lengths of solution paths formed by the ACO+GA algorithm are shown in the Figure 2.

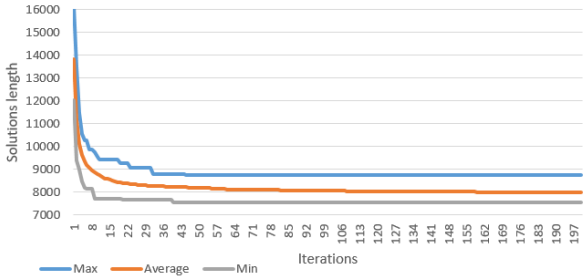

Figure 2: Length of paths founded by the ACO+GA algorithm

The maximum, minimum, and average lengths of solution paths formed by the SACO algorithm with optimal parameters are shown in the Figure 3.

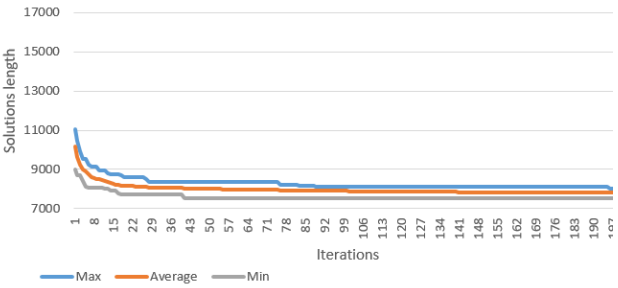

Figure 3: Length of paths founded by the SACO algorithm with optimal parameters

The maximum, minimum, and average lengths of solution paths formed by the SACO algorithm with not optimal parameters are shown in the Figure 4.

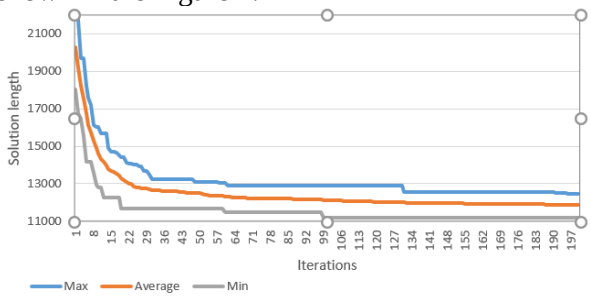

Figure 4: Length of paths founded by the SACO algorithm with not optimal parameters

Analyzing the graphs, one can see that the spread of values in the SACO+GA algorithm is not as large as in SACO with not optimal parameters. However, the spread of SACO with optimal parameters is much lower.

Comparison of the average lengths of solution paths formed by algorithms is shown in the Figure 5.

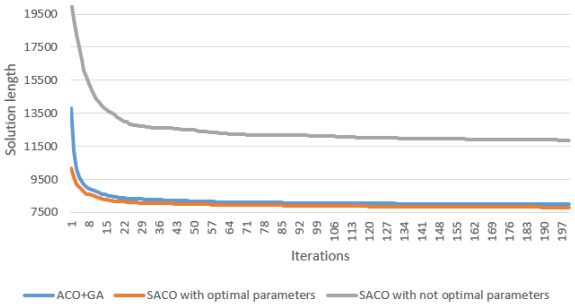

Figure 5: Comparison of the average lengths of solution paths

Analyzing the results, we can say that the proposed algorithm is stable to the initial parameters and significantly increases the quality of the search. In the best case, the algorithm finds the optimal solution, as well as the SACO algorithm with optimal parameters. On average, the algorithm works 34% better than SACO with not optimal parameters. Also on average, the algorithm works 2% worse than SACO with optimal parameters.

The algorithm shows results not much worse than SACO with optimal parameters. However, the algorithm becomes less susceptible to parameters and more versatile for solving combinatorial problems of any configuration.

6. Conclusions

The proposed ant colony optimization with parameter update using a genetic algorithm presented in this paper is shown to produce better results for the solution of traveling salesman problems. The algorithm becomes insensitive to the original parameter values and produces results not much worse than with optimal parameters.

Actually, this ACO+GA is a promising approach which can be used for different problems. The main goal is achieved; the algorithm has become less susceptible to parameters.

We showed the basic idea of how to work with parameters. The researcher can use a different algorithm to optimize the parameters or improve the proposed one.

However, it begs the question of how sensitive the new algorithm is to the parameters of the genetic algorithm. Intuitively, we can assume that the sensitivity is low, since the parameters affect the search for ACO parameters, and not the search for a solution. Nevertheless, it can be a promising research topic.

7. Acknowledgments

This work is supported by the Russian Foundation for Basic Research (project 20-01-00610).

References

[1] M. Hassan, M. Hasan, M. M .A. Hashem, An improved acs algorithm for the solutions of larger tsp problems, arXiv:1304.3763 (2013).

[2] J.R. Batmetan, A. Santoso, Multiple-objective ant colony algorithm for optimizing disaster relief logistics, Advanced Science Letters 23, no. 3 (2017). doi:10.1166/asl.2017.8758

[3] D. Liang, Zh. Zhan, Y. Zhang, J. Zhang, An Efficient Ant Colony System Approach for New Energy Vehicle Dispatch Problem, IEEE Transactions on Intelligent Transportation Systems (2019). doi: 10.1109/tits.2019.2946711.

[4] K. Zhou, W. Yongzhao, W. Wanying, N. Zhiyong, J. Tianguo, L. Xiaojun, Cloud Service Optimization Method Based on Dynamic Artificial Ant-Bee Colony Algorithm in Agricultural Equipment Manufacturing, Mathematical Problems in Engineering 2020 (2020). doi: 10.1155/2020/9134695.

[5] X. Qian, X. Zhong, Optimal individualized multimedia tourism route planning based on ant colony algorithms and large data hidden mining, Multimedia Tools and Applications 78, no. 15 (2019): 22099-22108.

[6] ZJ. Lee, S.F. Su, C.C. Chuang, KH Liu, Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment, Applied Soft Computing 8, no. 1 (2008): 55-78..

[7] O. Castillo, H. Neyoy, J. Soria, P. Melin, F. Valdez, A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot, Applied soft computing 28 (2015): 150-159. doi: 10.1016/j.asoc.2014.12.002

[8] Z. Yan-hua, F. Lei, Y. Zhi, Optimization of cloud database route scheduling based on combination of genetic algorithm and ant colony algorithm, Procedia Engineering 15 (2011): 3341-3345.

[9] Filimonov, S. Vakhrushev,M. Wurz, A. Shaganov, L. Rissing, Investigation of the crystallization of NiFe81/19 depending on the annealing temperature. ECS Transactions (2013): 147.

[10] TA. El-Mihoub, AA Hopgood, N Lars, A Battersby, Hybrid genetic algorithms: A review (2006).

[11] B. Koçer, The Analysis of GR202 and Berlin 52 Datasets by Ant Colony Algorithm, In 2015 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT), IEEE (2015):103-108. doi: 10.1109/ACSAT.2015.47