Computer Science • 25(3) 2024 https://doi.org/10.7494/csci.2024.25.3.6360

Krzysztof Ostrowski
Grażyna Starzec
Mateusz Starzec

THE ANT COLONY OPTIMIZATION ALGORITHM APPLIED IN TRANSPORT LOGISTICS

Abstract *The Vehicle Routing Problem belongs to graph optimization and its goal is to find shortest routes visiting a given set of customers with additional constraints present. The article presents the ant colony optimization metaheuristic which solves vehicle routing problems and its real-life application in transport logistics (finding routes for delivery companies). The metaheuristic generated high-quality solutions (superior to compared methods). Our tool is flexible and enables us to solve various variants of routing problems so it is well suited to specific needs of transportation companies.*

Keywords ant colony optimization, ACO, metaheuristic, routing problems, transport logistics, delivery

Citation Computer Science 25(3) 2024: 1–20

1. Introduction

Routing in transportation networks is a daily task for transport industry companies (i.e. transportation, delivery planning or waste collection). Constructing efficient routes can be challenging (especially for larger fleets of vehicles and with multiple additional constraints) but is necessary to minimize costs (i. e. fuel consumption and work time). Routes optimization can be regarded as a part of green logistics, which is an activity aiming to reduce the environmental impact of companies logistics. Constructing efficient routes that visit a given set of customers usually have additional constraints i. e. associated with vehicles capacity, customers opening-times or vehicles distance/work-time limit. Optimization problems arising from practical needs of transportation companies belong to the family of Vehicle Routing Problems (VRP). The problem's computational difficulty (impractical execution time of exact methods for larger instances) triggered the search for efficient metaheuristic solutions.

The authors' previous research was focused mostly on metaheuristics and graph problems [23, 27]. The authors propose the ant-colony optimization metaheuristic (ACO) as a part of a web application, which solves various problems from VRP family (dependent on the needs of transportation companies). The presented algorithm is a combination of techniques and variants of ACO and local optimizations. This is a result of comprehensive tests conducted on tens of real-life optimization tasks. We collected this data from companies that are interested in implementation of our solution in their daily work. Our work was associated with a project named "TRASA – development and validation of algorithms for routes optimization and resources allocation" [13]. The article is organized as follows: Section 2 presents VRP family, Section 3 includes mathematical formulations, Section 4 outlines some existing solutions, Section 5 describes the algorithm, Section 6 presents real-life results and conclusions are included in Section 7.

2. VRP family

The Vehicle Routing Problem is a multiple route generalization of the well-known Travelling Salesman Problem (TSP) [5], where the goal is to find a Hamiltonian cycle (a path in a graph visiting all vertices), which minimizes the total cost of visited edges. VRP is a family of combinatorial optimization problems, which has many real-life applications (i. e. those mentioned above) and dates back to 1959 [9]. In the basic version of the problem there is a graph of n customer locations and a fleet vehicle consisting of m cars. There is also one or more depots: special locations representing start and end of cars routes. The graph is weighted (costs associated with edges between locations). The goal is to visit all customers (using at most m cars) while minimizing the total cost of all routes (each route starts and ends in given depots).

VRP has many variants, which add constraints to the basic optimization problem and were defined to address practical routing problems. In Capacitated Vehicle Routing Problem (CVRP) [15] each vehicle has a limited capacity for goods that are

delivered to clients (each client has some demand) and therefore routes sizes are limited by max. capacity. In this case, number of vehicles used can be also optimized (as the first optimization criterion - the second being total routes cost). In Vehicle Routing Problem with Time Windows (VRPTW) each client has a time-window [7]. In hard TW version goods can be delivered only in this time slot: arriving too early means waiting for a TW opening while arriving too late means delivery is impossible. In VRP versions with soft time-windows (VRPSTW) delivery time bounds can be violated but at a penalty [4]. In time-window versions of the problem edge also have assigned travel times and there are also service times assigned to each client. In Distance Constrained Capacitated Vehicle Routing Problems (DCVRPs) the duration of each vehicle route (defined as the sum of travel time, service times and depot load/unload time) is limited [2]. Depending on version, capacity constraints can be applied to the whole vehicle route or only to a route section between subsequent depots (if we allow for vehicle's multiple comebacks to depots, where it is unloaded). In the Time-dependent Vehicle Routing Problem (TDVRP) travel times between clients depend on the moment of travel start. This problem version can model traffic in road networks (i.e. lower speeds in peak hours) [12]. In stochastic versions of VRP clients demands or travel times are not known beforehand (they are random variables) [20].

In the article we deal mostly with VRP variants including capacity, drivers work time and time-windows constraints: they can be classified as DCVRPTW (Distance Constrained Capacitated Vehicle Routing Problems with Time Windows). Both strict time-windows as well as soft time-windows versions are implemented in our algorithm.

3. Problem formulation as mixed-programming problem

The definition of DCVRPTW with strict time windows we use here is based on the flex version of DCVRP [2] (multiple comebacks to depots are allowed) but with additional time-windows constraints added. Let K be the set of vehicles. Let N be the set of customers, H_0 the set of original depots, $H_1 ... H_{B-1}$ sets of intermediate depots (which vehicles can come back to inside their routes), H_B the set of final depots and V the set of all nodes (the union of all previously defined sets). Let d_i be nonnegative demand of customer $i \in N$. Let c_{ij} be the cost of travelling from node i to node j and t_{ij} its travel time. Let s_i be service time of customer i. Let o_i be time-window opening of customer i and cl_i its closing time. Let Q be the maximal vehicle load while R be the maximal vehicle route time. Let x_{ijk} be a binary variable indicating if vehicle k travels from node i to node j. Let e_{ik} be an auxiliary, binary variable indicating if customer i is visited by vehicle k. It is defined as $\sum_{j \in V} x_{ijk}$. Let y_k be a binary variable indicating if vehicle k is used in the solution. Let z_{ik} be the total load served by vehicle k between its last visit to depot and visit to node i (including the load of node i). It is set to 0 at depot nodes. Let w_{ik} be the arrival time of vehicle k at node i. It is initialized to 0 at original depot nodes (that start vehicle's route).

The goal is to minimize formula (1)

$$\sum_{i \in V} \sum_{j \in V \setminus \{i\}} \sum_{k \in K} x_{ijk} c_{ij} + \sum_{k \in K} y_k m \tag{1}$$

subject to constraints:

$$\sum_{j \in V \setminus \{i\}} \sum_{k \in K} x_{ijk} = 1 \quad \forall_{i \in N} \tag{2}$$

$$\sum_{i \in V \setminus \{j\}} x_{ijk} - \sum_{i \in V \setminus \{j\}} x_{jik} = 0 \quad \forall_{j \in N, k \in K} \tag{3}$$

$$\sum_{i \in H_0} \sum_{j \in N} x_{ijk} - y_k = 0 \quad \forall_{k \in K} \tag{4}$$

$$\sum_{i \in H_{b-1}} \sum_{j \in N} x_{ijk} - \sum_{i \in H_b} \sum_{j \in N} x_{ijk} = 0 \quad \forall_{k \in K, H_b \in \{H_1, \ldots, H_B\}} \tag{5}$$

$$\sum_{i \in N} x_{ijk} - \sum_{i \in N} x_{jik} \geq 0 \quad \forall_{j \in \{H_1, \ldots, H_{B-1}\}, k \in K} \tag{6}$$

$$x_{ijk} = 0 \quad \forall_{i \in H_B, j \in N, k \in K} \tag{7}$$

$$(z_{ik} + d_j - z_{jk}) \leq M(1 - x_{ijk}) \quad \forall_{i \in V, j \in N \setminus \{i\}, k \in K} \tag{8}$$

$$(z_{ik} + d_j - z_{jk}) \geq -M(1 - x_{ijk}) \quad \forall_{i \in V, j \in N \setminus \{i\}, k \in K} \tag{9}$$

$$(w_{ik} + s_i + t_{ij} - w_{jk}) \leq M(1 - x_{ijk}) \quad \forall_{i \in V, j \in \{N, H_1, \ldots, H_B\} \setminus \{i\}, k \in K} \tag{10}$$

$$(w_{ik} + s_i + t_{ij} - w_{jk}) \geq -M(1 - x_{ijk}) \quad \forall_{i \in V, j \in \{N, H_1, \ldots, H_B\} \setminus \{i\}, k \in K} \tag{11}$$

$$0 \leq z_{ik} \leq Q \quad \forall_{i \in N, k \in K} \tag{12}$$

$$0 \leq w_{ik} \leq R \quad \forall_{i \in \{N, H_1, \ldots, H_B\}, k \in K} \tag{13}$$

$$e_{ik} \cdot o_i \leq e_{ik} \cdot w_{ik} \leq e_{ik} \cdot cl_i \quad \forall_{i \in N, k \in K} \tag{14}$$

$$z_{ik} = 0 \quad \forall_{i \in \{H_0, \ldots, H_B\}, k \in K} \tag{15}$$

$$w_{ik} = 0 \quad \forall_{i \in H_0, k \in K} \tag{16}$$

$$x_{ijk} \in \{0, 1\} \quad \forall_{i,j \in V, k \in K, i \neq j} \tag{17}$$

$$y_k \in \{0, 1\} \quad \forall_{k \in K} \tag{18}$$

In Equation (1) constant m set to 0 means that only total routes cost is minimized while setting it to a sufficiently large value will minimize number of cars used and then total cost. Constraint (2) means that each customer must be visited exactly once by exactly one vehicle. Equation (3) is a flow conversion constraint (vehicle arriving at a given customer node has to leave it afterwards). Equation (4) is a relation between variables x and y. Equations (5) and (6) are conversion of flow through depots constraints. Constraint (7) eliminates any flow outcoming from final depots. Constraints (8) and (9) define relations between load variables and customer demands.

Constraints (10) and (11) define relations between time variables, travel times and customers service times. In those constraints we assume sufficiently large constant M (i.e. larger than the sum of client demands in Equations (8) and (9)). Constraint (12) represents the load limits and constraint (13) the route time limits. Constraint (14) forces time window obedience but only for customers visited by vehicle k (therefore e variable is introduced in the formula). Constraints (15) and (16) reset load and time variables in depots (in all depots in case of load and in starting depots in case of time). Constraint (17) and (18) forces variables x and y to be binary.

In DCVRPTW with soft time windows we allow vehicles to arrive after customers closing time. Equation (14) from the above definition is replaced with

$$e_{ik} \cdot o_i \leq e_{ik} \cdot w_{ik} \qquad \forall_{i \in N, k \in K} \tag{19}$$

In this variant we also try to minimize the total time-windows delay of all visited customers. A delay is non-zero for a customer visited after its TW close time and is defined as the difference between vehicle arrival time and TW close time. Therefore it is a bi-objective optimization with two optimization criteria (for total cost and total time windows delay):

$$\sum_{i \in V} \sum_{j \in V \setminus \{i\}} \sum_{k \in K} x_{ijk} c_{ij} + \sum_{k \in K} y_k m \tag{20}$$

$$\sum_{i \in N, k \in K} e_{ik} \cdot max(0, w_{ik} - c_i) \tag{21}$$

4. State of the art

The VRP is an NP-hard optimization problem [18] and no polynomial-time exact algorithms are known. Branch-and-cut solutions [3] are among most effective exact approaches for problems from TSP and VRP family. However, for large problem instances their computation time is usually impractically long. Therefore the main focus of researchers has been on heuristics and metaheuristics, which can find satisfactory solutions (but usually not optimal) in shorter execution time. One of the first heuristic for VRP was savings algorithm [8], where initially each route includes only one customer and routes are merged according to a criterion maximizing distance reduction until no further merges are feasible regarding the problem constraints. Metaheuristics have been successfully applied to solve many optimization problems and they include exploration (discovering new regions in solutions space) and exploitation (searching promising regions intensely) phases. Some of them operate on single solutions (i.e. iterated local search (ILS) [1] or tabu search (TS) [11]) and some of them maintain a population of solutions: i.e. nature-inspired methods like evolutionary algorithms (EA) [24], particle swarm optimization (PSO) [16] and ACO metaheuristics. Some researchers also tried the combination of state-of-the art metaheuristics with machine learning [25, 26].

5. Algorithm description

The authors proposed the ACO metaheuristic [10] to solve routing problems. This approach is inspired by the behaviour of ants and is well suited to solve graph optimization problems. It is a probabilistic, multi-agent approach, which belongs to swarm intelligence methods. Each ant (an agent) moves between various states and constructs a solution. The probability of a transition between states depends on *a priori* knowledge (desirability) and on pheromone levels (swarm intelligence – knowledge gathered during the algorithm search). The procedure of solutions construction is repeated multiple times allowing the colony to learn more about the problem instance. In graph optimization problems each ant traverses edges until a feasible solution is constructed, desirability is directly associated with edge costs (i.e. travel time or distance) while edges pheromone levels are updated according to the quality of solutions they form. Due to the fact that each ant constructs its solution independently ACO parallelization can be efficiently implemented [19,28], which is also an important quality of this metaheuristic. In addition, the specificity of ACO solution construction makes adaptation to various optimization targets and creation a flexible tool solving real-life problems easier. The algorithm pseudocode is shown at the end of this section.

5.1. Solution construction

The algorithm consists of n main iterations. In every iteration each of k ants builds a solution beginning from a given starting vertex. To preserve solutions feasibility (regarding constraints like max. work time, time-windows or car capacity) ants can only choose edges that do not cause constraint violations. The probability of choosing edge ij is given by the following formula:

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{ik \in feasibleMoves} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}} \tag{22}$$

In the above formula η_{ij} represents desirability of a given edge, which is inversely proportional to its cost (distance or travel time, depending on optimization criteria), τ_{ij} represents colony knowledge regarding a given edge, while α and β parameters control the influences of colony knowledge and desirability knowledge.

If a solution being built is still not complete and no new vertex can be visited without violating constraints then an ant goes back to the depot vertex and a new subpath is constructed (in a new work time slot, if necessary).

5.2. Local optimization

After a feasible solution is build it undergoes a series of local optimizations. ACO performs an exploration of solution space and finds promising regions while local search methods enable for a better exploitation of those regions. Local search operators include 2-opt, insert and delete methods.

5.2.1. 2-opt

Two-opt is an operator that chooses two non-adjacent edges in a path: $(p_i p_{i+1})$ and $p_j p_{j+1}$ and replaces them with edges $(p_i p_j)$ and $(p_{i+1} p_{j+1})$. In other words, it inverts a given path fragment $(p_{i+1}...p_j)$. In our solution we apply a hill climbing version, which modifies a path as long as there exists an improving move (regarding our optimization criterion or criteria). The operator is presented in Figure 1.

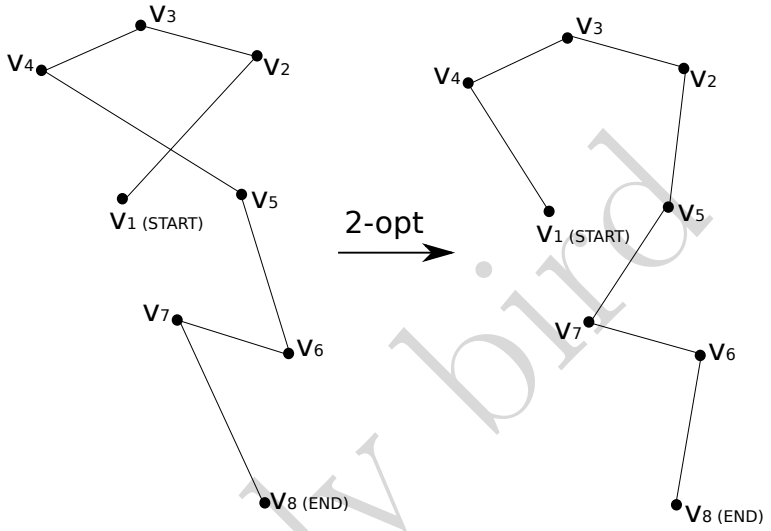

Figure 1. An example of 2-opt local search. First move inverts path fragment v2-v3-v4 and the second move inverts fragment v6-v7. As a result a local optimum is reached

5.2.2. Insert and delete

Another local search procedure used involves insert and delete operators. After deleting a given group of vertices (they can be chosen randomly or according to a heuristic) they are inserted back into a solution in such places that the solution cost increase is minimized. This procedure can be applied multiply for varying number of modified vertices until no further improvement is found.

5.2.3. Generalized move – path look improvement and final optimization

The best final paths undergo an additional local search, which moves groups of points between paths in order to maximize a certain criterion. In commercial applications, besides meeting optimization goals, a visual aspect of solutions can be important (how clients perceive visualization of solutions). For example it is visually better if one route serves mostly a certain area/city district without serving too many clients from other districts. Normally even high-quality solutions do not guarantee this as clients from various districts are often on vehicles' way to other areas. This operator

can improve compactness of paths without deteriorating solutions quality. It is based on mean square error (MSE) measure known from clustering.

Alternatively, this operator can serve as the last optimization of the resulting solution (improving distance instead of MSE) if the only goal is to optimize path cost.

5.3. Pheromones update

After solutions are constructed and optimized it is time for pheromones update. Initially, all pheromones undergo the procedure of evaporation (their value decrease at the end of an iteration). Afterwards, the best solutions (from the current iteration and from the whole algorithm run) are used and the edges they consist of have pheromone levels increased. In this way the colony updates its knowledge regarding the solved problem instance. The update procedure is given by the following formula:

$$\tau_{ij} \leftarrow \tau_{ij} \cdot (1 - \rho) + l \cdot \gamma \tag{23}$$

In the formula ρ is an evaporation coefficient, γ is an update coefficient and l is the number of solutions (among the best ones) including edge ij. Parameters ρ and γ control the rate of colony knowledge change. The bigger those values the faster knowledge update takes place. Pheromone amounts on edges are limited to an interval $[\tau_{min}, \tau_{max}]$ as it is in the min-max ant system (MMAS).

5.4. Construction of pareto set approximations

In some problem versions, more than one optimization criteria is applied. Each solution can be then described by a criterion vector $(y_1, y_2, ..., y_n) \in R^n$. Let $y' \succ y''$ mean that solution y' dominates solution y''. The definition of domination is as follows:

$$y' \succ y'' \iff \forall_{1 \leq i \leq n} \, (y'_i \leq y''_i) \wedge \exists_{1 \leq i \leq n} \, (y'_i < y''_i) \tag{24}$$

This means that y' is not worse than y'' in any optimization criterion but is better in at least one (assuming a minimization problem). A set of all non-dominated solutions is called pareto front. The algorithm maintains a set of pairwise non-dominated solutions (approximation of the pareto front). Let Y be the set of all feasible solutions (their optimization criteria results). Pareto front P definition is given below:

$$P = \{y \in Y : \neg \, \exists_{y' \in Y} (y' \succ y)\} \tag{25}$$

In case of multiple criteria optimization each ant has pseudorandom weights assigned (from $[0, 1]$ interval), which signal the importance of each optimization target. Then pheromones and desirability computations are based on those weights and optimization targets properties. Let $(\eta_1, ..., \eta_n)$ and $(\tau_1, ..., \tau_n)$ represent vectors of desirability and colony knowledge for all optimization targets and for one particular move and let $(w_1, ..., w_n)$ be a vector of optimization target weights of a given

ant. Then composite desirability and colony knowledge for all optimization targets is calculated as follows:

$$\tau = \prod_{1 \leq i \leq n} (\tau_i \cdot w_i) \tag{26}$$

$$\eta = \prod_{1 \leq i \leq n} (\eta_i \cdot w_i) \tag{27}$$

Pseudorandom distribution of weights among ants lets the algorithm to approximate various fragments of the pareto front.

Algorithm 1 ACO

```
1: bestSolutions = φ
2: for i ⇐ 1...n do
3:     solutions = φ
4:     for j ⇐ 1...k do
5:         currentSolution = (startVertex)
6:         while currentSolution is not complete and there are feasible moves do
7:             choose a vertex w (probability based on desirability and pheromones)
8:             append vertex w to currentSolution
9:         end while
10:        optimize locally currentSolution
11:        add currentSolution to solutions
12:    end for
13:    update bestSolutions with solutions
14:    update pheromones based with top solutions from solutions and bestSolutions
15: end for
16: return bestSolutions
```


6. Experiments

The algorithm was applied to solve practical optimization problems for a few transportation companies. Our tool is flexible and enables us to solve various versions of VRP family (regarding constraints and optimization targets) and therefore it is well suited to meet specific needs of the companies. Experiments were conducted on a computer with 3.5 GHz processor (4 cores), 16 GB of RAM and Linux operation system. The application was written in Scala programming language. The algorithm's parameters values are given in Table 1. Real distances and approximate travel times between client addresses were obtained from Open Route Service [21]. For comparison purposes we decided to use OR optimization tool [22] for some companies test instances. We compared our method with Greedy Descent (GD) and Guided Local Search (GLS) metaheuristics [30]. The metaheuristics time limit was set to the same values as ACO execution times. In addition, we executed our algorithm on VRPTW benchmarks with known optimal solutions (Solomon instances) and compared with

other metaheuristics. The results presented are average of multiple runs (to obtain some statistics) but usually in our web application one run is performed (it can be reconfigured to choose the best of n runs too).

Table 1
ACO parameters values

Parameter	Name	Value
n	Iterations count	150
k	Ants count	25
α	Colony knowledge coefficient	20
β	Desirability coefficient	40
ρ	Evaporation coefficient	0.01
γ	Pheromone update coefficient	0.03

6.1. Parameters tuning and analysis

Iterations count and ants count were set in a way to reach high-quality solutions in reasonable execution times. For a given ants number iterations count was determined empirically. In Table 2 there is a comparison of algorithm results for various ants and iterations count. Due to increase of exploration power ACO generates better results with ants count increase. We chose 25 ants as for larger count the increase of solution quality is getting smaller while execution time grows significantly. In Figure 2 an exemplary algorithm run is presented (iterations vs best result). The result before final local optimization (in post-ACO phase) is given.

Table 2
Tuning results for ants count – results for catering company network (Gdansk)

ants	iterations	result	% gap	execution time
10	75	440.8	2.8	19
15	100	436.2	1.7	35
25	150	428.9	-	78
40	250	426.7	-0.5	207

To finetune the remaining parameter values of the ACO we ran the algorithm with four possible values of α (10, 20, 40, 80), four values of β (10, 20, 40, 80), three values of ρ (0.01, 0.03 and 0.1) and three values of γ (0.01, 0.03 and 0.1). Therefore, 144 combinations of parameter values were tested on two instances (Catering company instance – Gdansk and one of Solomon VRPTW benchmark instances). For each combination in 4-dimensional space of parameter values ACO was run 10 times for both test instances and we computed average gap (to the optimal solution in case of Solomon instance and to the best known solution in case of Gdansk instance). In this way we determined the best set of parameter values.

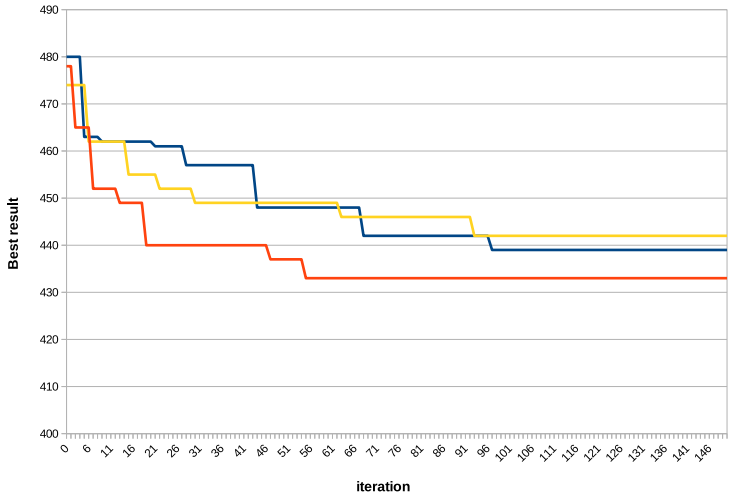

Figure 2. Exemplary algorithm runs: best result so far vs iteration number, Gdansk network

In Table 3 we present averaged tuning results for slow evaporation and moderate pheromone update rate configuration. It can be seen that configurations with the smallest utilization of the colony knowledge ($\alpha = 10$) produced the worst result while there are minor differences between configurations varying in desirability coefficient. The best configuration has moderate values of desirability and pheromone coefficients and most of them are within 0.5 percent of the best one. Small differences indicate the algorithm's robustness regarding parameter values change and can be also associated with strong influence of local search procedures on the final result (they exploit solution space areas searched by the metaheuristic).

Table 3

Results of parameter tuning for slow evaporation ($\rho = 0.01$) and moderate pheromone update rate ($\gamma = 0.03$). Average results for given α and β values as well as the best configuration are in bold. Combined average gap of both tuning test instances is presented

α/β	10	20	40	80	avg.
10	2.5	2.8	2.0	1.9	**2.3**
20	1.2	1.5	**1.0**	1.6	**1.3**
40	1.2	1.3	1.4	1.5	**1.3**
80	1.2	1.4	1.3	1.7	**1.4**
avg.	**1.5**	**1.8**	**1.4**	**1.7**	–

In Table 4 tuning results for moderate evaporation and fast pheromone update rate are presented. Generally, differences between various α and β value sets are smaller than in the previous configuration. With faster pheromone evaporation and update even configurations with smaller colony knowledge utilization ($\alpha = 10$) have reached high-quality solutions.

Table 4

Results of parameter tuning for moderate evaporation ($\rho = 0.03$) and fast pheromone update rate ($\gamma = 0.1$). Average results for given α and β values as well as the best configuration are in bold. Combined average gap of both tuning test instances is presented

α/β	10	20	40	80	avg.
10	2.0	**1.2**	1.3	2.0	**1.6**
20	1.4	1.2	1.4	1.6	**1.4**
40	1.9	1.5	1.2	1.8	**1.6**
80	1.4	1.3	1.7	2.3	**1.7**
avg.	**1.6**	**1.3**	**1.4**	**1.8**	-

In Table 5 there are tuning results for high evaporation and moderate pheromone update rates. This time the best results are reached for smaller α values. It seems that increasing evaporation and update rates compensates for a lower pheromone trail coefficient. On the other hand, the algorithm consistently generates worse results for the highest value of desirability coefficient ($\beta = 80$). This can be associated with extensive local search methods, which already utilize the problem-specific knowledge so there is no need to overuse the heuristic component during paths construction.

Table 5

Results of parameter tuning for high evaporation ($\rho = 0.1$) and moderate pheromone update rate ($\gamma = 0.03$). Average results for given α and β values as well as the best configuration are in bold. Combined average gap of both tuning test instances is presented

α/β	10	20	40	80	avg.
10	2.0	**1.2**	1.3	2.0	**1.6**
20	1.8	1.7	1.9	2.0	**1.8**
40	1.6	1.9	1.9	2.1	**1.9**
80	1.8	1.9	2.6	3.1	**2.3**
avg.	**1.8**	**1.7**	**1.9**	**2.3**	-

In Table 6 tuning results for various pheromone evaporation and update rates are presented (global average). It can be seen that the best results are generated when update rate is three times higher than evaporation rate. The differences are small but higher values of update rates generally are associated with better average results.

Table 6

Tuning results for various values of pheromone evaporation and update rates averaged over all values of α and β. The best configurations in bold

ρ/γ	0.01	0.03	0.1
0.01	2.0	**1.6**	1.8
0.03	2.0	2.0	**1.5**
0.1	1.8	1.9	1.9

6.2. Catering company results

The algorithm was applied to help a Polish catering company with their route planning in Polish cities and regions (Warsaw, Gdansk and north-eastern Poland). The transport of goods was scheduled for one night. Clients time windows (of sizes: 4–11 hours) were the main constraint on vehicles' routes size. No time limit on routes exists in this instance. There were 1539 clients in Warsaw and 16 vehicles available. In NE Poland there were 2117 clients and 29 vehicles available. In Gdansk there were 345 clients and 4 vehicles available. The goal of the optimization was to minimize the total distance covered by all vehicles without violating time-windows constraints. We tested networks for two versions of the algorithm: a commercial variant (improving paths look as final optimization) and performance variant (further distance improvement in the final optimization). The results are given in Table 7. It can be seen that the version which incorporates further distance improvement is on average 4–5 percent better than the commercial version. In Table 8 there is a comparison of ACO with metaheuristics from the OR tool. ACO clearly outperforms other metaheuristics for Warsaw and NE Poland data sets. Average advantage over GLS is 3.2 percent and over GD 4.8 percent.

Table 7

Catering company results. Results are the average of 30 algorithm runs. The version with path look improvement is marked as ACO_P while the version with final distance improvement is marked as ACO_D. Solution distances are given in kilometers while execution times in seconds

	ACO_P			ACO_D		
Instance	Vehicles no	Distance	Ex. time	Vehicles no	Distance	Ex. time
Warsaw	12	1231.7	108	12	1211.3	131
Gdansk	4	437.4	70	4	428.1	78
NE Poland	28	7388.7	650	28	7180.4	755

Table 8

Comparison of ACO with GD and GLS (catering company results). Results are the average of 30 runs. Confidence intervals (alfa = 0.05) given in +/- column

	ACO_D		GD			GLS		
Instance	Distance	+/–	Distance	+/–	% Gap	Distance	+/–	Gap
Gdansk	428.1	1.0	460.3	3.9	7.5	442.8	3.7	3.4
Warsaw	1211.3	5.7	1210.8	4.3	0.0	1223.3	5.1	1.0
NE Poland	7180.4	35.1	7675.4	70.6	6.9	7558.2	63.2	5.2
Avg.	–	–	–	–	**4.8**	–	–	**3.2**

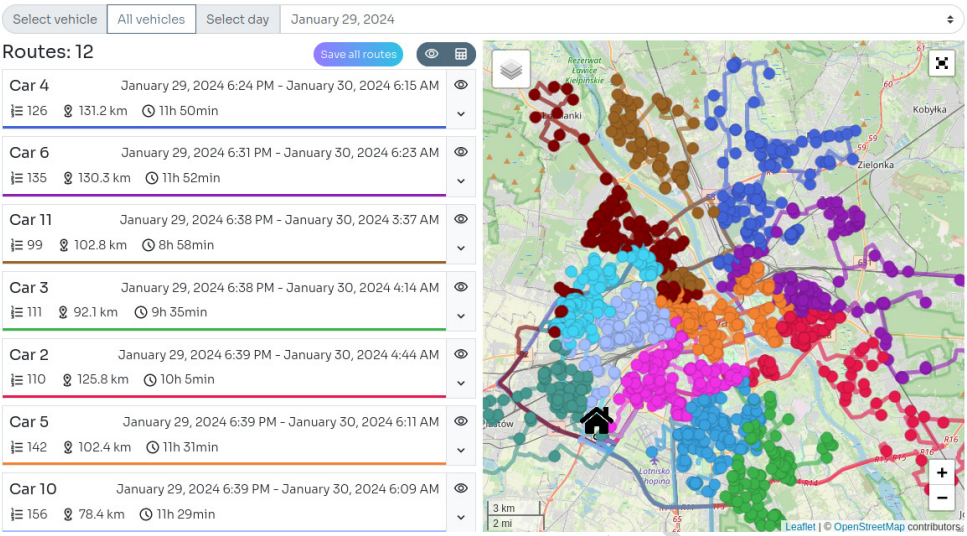

Figure 3. Algorithm run results for the catering company displayed in the application window (for the city of Warsaw)

6.3. Pharmaceutical wholesaler company results

The algorithm was applied to generate solutions for a pharmaceutical company. In one version a depot was located in Krakow (301 client addresses to visit and 15 vehicles) and in the other version in Rzeszow (131 client addresses to visit and 8 vehicles). There were two distinct capacity limits: one was a standard truck capacity and the other was a maximal capacity for cold products (that have to be transported in low temperatures). Drivers work hours were 6–18 (12 hour route time limit) for 5 days a week but some of them did not have a fixed limit. Therefore, this is a generalization of DCVRPTW, where drivers work hours replace standard route time limit. In this version there are soft time windows and one of the objectives was to minimize total violation of time windows (some clients were served later than its time-window close time). Therefore we have two optimization targets (total time windows delay and total distance). In this configuration, depending on clients flexibility, one can choose any of the generated solutions in our application (clients TWs were usually narrow and mostly in the first half of the day).

In Figure 4 the results of ACO bi-objective optimization are compared with GLS and GD algorithms. To generate various solutions in the OR tool (which has one optimization target) a few values of penalties (for exceeding soft time window upper bound) were used and each configuration was executed a few times. ACO outperformed other metaheuristics regarding total TW delay reaching the lowest values on pareto front approximations. All three metaheurisics were very similar regarding total distance (GD had on average 1 percent longer routes than GLS and ACO).

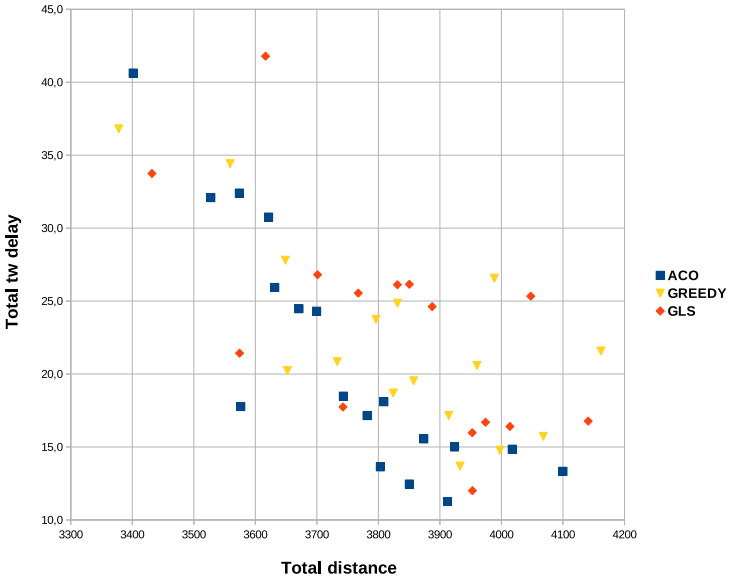

Figure 4. Comparison of ACO, GLS and GD metaheuristics results, bi-objective optimization for the pharmaceutical company (depot in Krakow)

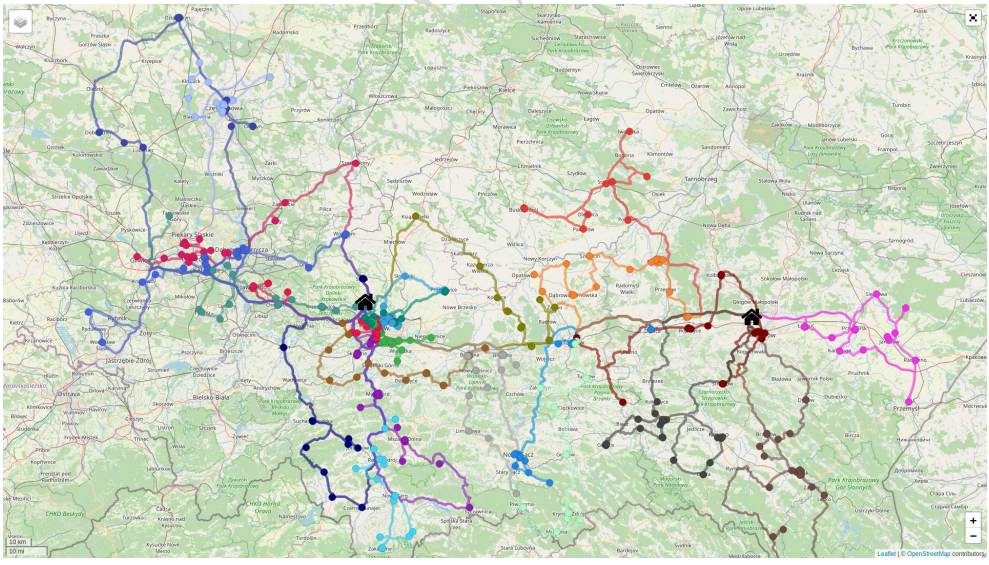

Figure 5. Algorithm run results for the pharmaceutical company (Krakow and Rzeszow combined results)

6.4. Delivery company results

We also planned routes for a delivery company. The goal was to plan delivery in Warsaw and surrounding towns. There were 5090 clients and 116 available vehicles. Deliveries were planned for one day and the work time limit for every vehicle was 10 hours. It was a multi-objective optimization as there were two optimization targets: total distance and total work time. The best results by each objective are presented in Table 9 The approximate pareto-front is not wide due to both criteria results being correlated with each other.

Table 9
Delivery company results. Results are the average of 10 algorithm runs

Criterion	Vehicles used	Total distance (km)	Total work time
Best by distance	104	10940.1	1005 h 35 min
Best by total work time	104	11130.2	1001 h 45 min

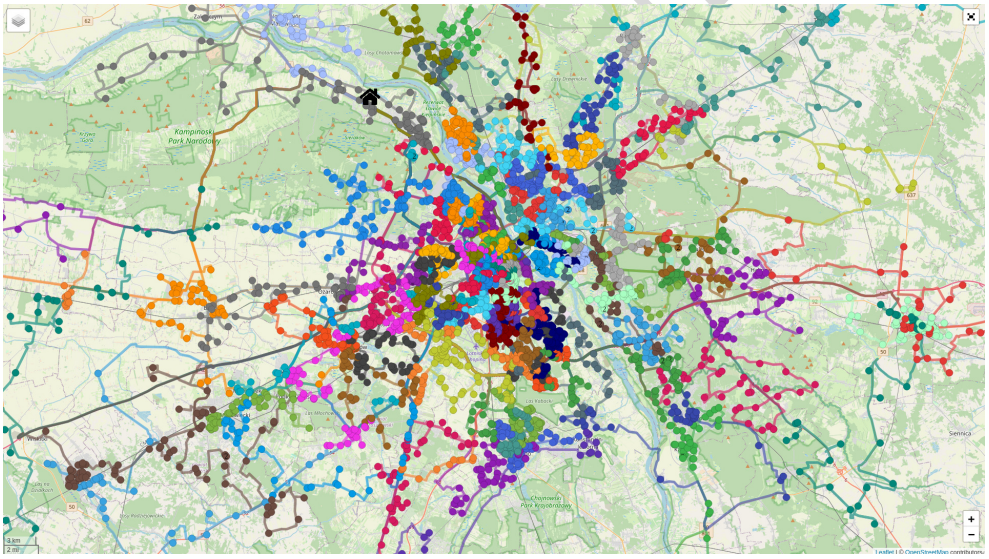

Figure 6. Algorithm run results for the delivery company (Warsaw and surroundings)

6.5. Benchmark instances results

To compare the solutions of our algorithm to known optimal solutions we executed it on some VRPTW benchmarks (Solomon C instances [31]). The results are presented in Table 10. It can be seen that the algorithm reaches optimal or nearly optimal results in short execution times. Only for two test instances the algorithm generated non-optimal solutions.

In Table 11 a comparison of ACO and other metaheuristics is presented. The compared metaheuristics are local search (LS) [6], tabu search (TS) [17], evolutionary algorithm (EVO) [14] and hybrid of TS and simulated annealing (HYB) [29]. Their results are among the best achieved by heuristic solutions for VRPTW instances. It can be seen that ACO produced the best results on average.

Table 10

Solomon C instances results. The average of 30 algorithm runs, standard deviations and gaps to optimal solutions are presented. Execution time is given in seconds

Instance	Vertex count	Vehicles count	Max. capacity	Distance	St. dev.	Gap %	Vehicles used	Execution time
C101	101	25	200	827.3	0.0	0.0	10	4.9
C102	101	25	200	827.3	0.0	0.0	10	2.1
C103	101	25	200	826.3	0.0	0.0	10	2.2
C104	101	25	200	828.7	2.9	0.7	10	2.1
C105	101	25	200	827.3	0.0	0.0	10	2.1
C106	101	25	200	827.3	0.0	0.0	10	1.9
C107	101	25	200	827.3	0.0	0.0	10	1.7
C108	101	25	200	827.3	0.0	0.0	10	2.0
C109	101	25	200	827.3	0.0	0.0	10	2.2
C201	101	25	700	589.1	0.0	0.0	3	1.8
C202	101	25	700	589.1	0.0	0.0	3	2.2
C203	101	25	700	588.7	0.0	0.0	3	2.4
C204	101	25	700	593.2	2.6	0.9	3	2.4
C205	101	25	700	586.4	0.0	0.0	3	2.0
C206	101	25	700	586.0	0.0	0.0	3	2.1
C207	101	25	700	585.8	0.0	0.0	3	2.0
C208	101	25	700	585.8	0.0	0.0	3	1.8

Table 11

Average results for Solomon C instances - comparison of the ACO and other metaheuristics.

Algorithm	C1 instances	C2 instances
ACO	827.3	588.1
LS	832.9	593.5
TS	832.1	589.9
EVO	828.4	589.8
HYB	841.9	612.4

7. Conclusion

In the paper we presented the ant-colony optimization metaheuristic for the Vehicle Routing Problems family. The algorithm can be applied to solve various variants of VRP and is a part of a web application, which meets practical needs of transport

industry companies. The algorithm obtained satisfactory solutions in acceptable execution times and had an advantage over other compared metaheuristics. The selection of the metaheuristic and local optimization methods provides good solution quality to execution time ratio. It is crucial for real-life applications, because quality gives operational savings for users, but the computation time is in some cases restricted by real-life operations and the time slot between the moment of collecting the last orders to be planned and the time when the drivers need to know their routes to prepare to departure.

Acknowledgements

Our work was done as a part of project "TRASA – development and validation of algorithms for routes optimization and resources allocation" [13] and was financed by Intelligent Development Operational Program 2014–2020, sub-program: Industrial Research and Development Projects Carried out by Enterprises.

References

[1] Álvarez A., Munari P.: Metaheuristic approaches for the vehicle routing problem with time windows and multiple deliverymen, *Gestão & Produção*, vol. 23(2), pp. 279–293, 2016.

[2] Alvina G.K., Ruey L.C., Quiang M.: Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots, *Mathematical and Computer Modelling*, vol. 47(1), pp. 140–152, 2008. doi: 10.1016/j.mcm.2007.02.007.

[3] Araque J.R., Kudva G., Morin T.L., Pekny J.F.: A branch-and-cut algorithm for vehicle routing problems, *Annals of Operations Research*, vol. 4, pp. 37–59, 1994. doi: 10.1007/bf02085634.

[4] Balakrishnan N.: Simple heuristics for the vehicle routing problem with soft time windows, *The Journal of the Operational Research Society*, vol. 44(3), pp. 279–287, 1993. doi: 10.1038/sj/jors/0440308.

[5] Bellmore M., Nemhauser G.L.: The Traveling Salesman Problem: a Survey, *Operation Research*, vol. 16, pp. 538–558, 1986.

[6] Braysy O.: Fast local searches for the vehicle routing problem with time windows, *Information Systems and Operations Research*, vol. 41, pp. 179–194, 2003.

[7] Braysy O., Gendreau M.: Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms, *Transportation Science*, vol. 39(1), pp. 104–118, 2005. doi: 10.1287/trsc.1030.0056.

[8] Clarke G.U., Wright J.W.: Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research*, vol. 12(4), pp. 568–581, 1964. doi: 10.1007/978-3-642-27922-5_18.

[9] Dantzig G., Ramser J.: The truck dispatching problem, *Management science*, vol. 6, pp. 80–91, 1959. doi: 10.1287/mnsc.6.1.80.

[10] Dorigo M., Stutzle T.: *Ant Colony Optimization.*, The MIT Press Cambridge, 2004. doi: 10.1109/ci-m.2006.248054.

[11] Gendreau M., Hertz A., Laporte G.: A Tabu Search Heuristic for the Vehicle Routing Problem, *Management Science*, vol. 40(10), pp. 1276–1290, 1994. doi: 10.1287/mnsc.40.10.1276.

[12] Gmira M., Gendreau M., Lodi A., Potvin J.Y.: Tabu search for the time-dependent vehicle routing problem with time windows on a road network, *European Journal of Operational Research*, vol. 288, pp. 129–140, 2021. doi: 10.1016/j.ejor.2020.05.041.

[13] *Grant "Trasa"*. Online: https://getsent.io/en/projects/trasa.

[14] Homberger J., Gehring H.: A two-phase hybrid metaheuristic for the vehicle routing problem with time windows, *European Journal of Operations Research*, vol. 62, pp. 220–238, 2005. doi: 10.1016/j.ejor.2004.01.027.

[15] Ibrahim A.A., Nassirou L., Rabiat O.A., Jeremiah A.I.: Capacitated Vehicle Routing Problem, *International Journal of Research – GRANTHAALAYAH*, vol. 7(3), pp. 310–327, 2019. doi: 10.29121/granthaalayah.v7.i3.2019.976.

[16] Jin A., Kachitvichyanukul V.: A Particle Swarm Optimisation for Vehicle Routing Problem with Time Windows, *International Journal of Operational Research*, vol. 4(4), pp. 519–537, 2009. doi: 10.1504/ijor.2009.027156.

[17] Lau H.C., Sim M., Teo K.M.: Vehicle routing problem with time windows and a limited number of vehicles, *European Journal of Operational Research*, vol. 148, pp. 559–568, 2003. doi: 10.1016/s0377-2217(02)00363-6.

[18] Lenstra J.K., Kan A.H.G.: Computational Complexity of Discrete Optimization Problems, *Annals of Discrete Mathematics*, vol. 4, pp. 121–140, 1979. doi: 10.1016/s0167-5060(08)70821-5.

[19] Menezes B., Herrmann N., Kuchen H., Neto F.: High-Level Parallel Ant Colony Optimization with Algorithmic Skeletons, *International Journal of Parallel Programming*, vol. 49, pp. 776–801, 2021. doi: 10.1007/s10766-021-00714-1.

[20] Niu Y., Shao J., Xiao J., Song W., Cao Z.: Multi-objective evolutionary algorithm based on rbf network for solving the stochastic vehicle routing problem, *Information Sciences*, vol. 609, pp. 387–410, 2022. doi: 10.1016/j.ins.2022.07.087.

[21] *Open Route Service*. Online: https://openrouteservice.org/.

[22] *OR tool*. Online: https://developers.google.com/optimization.

[23] Ostrowski K., Karbowska-Chilinska J., Koszelew J., Zabielski P.: Evolution-inspired local improvement algorithm solving orienteering problem, *Annals of Operations Research*, vol. 1, pp. 519–543, 2017.

[24] Prins C.: A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers and Operations Research*, vol. 31(12), pp. 1985–2002, 2004. doi: 10.1016/s0305-0548(03)00158-8.

[25] Pugliese L.D.P., Ferone D., Festa P., Guerriero F., Macrina G.: Combining variable neighborhood search and machine learning to solve the vehicle routing problem with crowd-shipping., *Optimization Letters*, pp. 1–23, 2022. doi: 10.1007/s11590-021-01833-x.

[26] Qi R., Li J.Q., Wang J., Jin H., Han Y.Y.: Qmoea: A q-learning-based multiobjective evolutionary algorithm for solving time-dependent green vehicle routing problems with time windows, *Information Sciences*, vol. 608, pp. 178–201, 2022. doi: 10.1016/j.ins.2022.06.056.

[27] Starzec M., Starzec G., Byrski A., Turek W.: Distributed Ant Colony Optimization Based on Actor Model, *Parallel Computing*, vol. 90(1), 2019. doi: 10.1016/j.parco.2019.102573.

[28] Stutze T.: Parallelization strategics for Ant Colony Optimization, *Lecture Notes in Computer Science*, vol. 1498, pp. 722–731, 1998.

[29] Tan K.C., Lee K.O.: Artificial intelligence heuristics in solving vehicle routing problems with time window constraints, *The Engineering Applications of Artificial Intelligence*, vol. 14, pp. 825–837, 2001. doi: 10.1016/s0952-1976(02)00011-8.

[30] Voudouris C., Tsang E., Alsheddy A.: *Handbook of Metaheuristics*, 2010.

[31] *VRPTW instances*. Online: http://vrp.galgos.inf.puc-rio.br/index.php/en/ (VRPTW benchmarks).

Affiliations

Krzysztof Ostrowski
Bialystok University of Technology, Faculty of Computer Science, ul. Wiejska 45A, 15-351 Bialystok, Sentio sp. z o.o., ul. Warszawska 6/32, 15-063 Bialystok, e-mail: k.ostrowski@pb.edu.pl

Grażyna Starzec
AGH University, Faculty of Computer Science, al. Adama Mickiewicza 30, 30-059 Krakow, Sentio sp. z o.o., ul. Warszawska 6/32, 15-063 Bialystok, e-mail: g.starzec@getsent.io

Mateusz Starzec
Sentio sp. z o.o., ul. Warszawska 6/32, 15-063 Bialystok, e-mail: m.starzec@getsent.io

Received: 13.06.2024
Revised: 15.06.2024
Accepted: 15.06.2024