Adaptive Ant Colony Optimization With Node Clustering for the Multidepot Vehicle Routing Problem

Petr Stodola and Jan Nohel

***Abstract*—This article deals with the novel metaheuristic algorithm based on the ant colony optimization (ACO) principle. It implements several novel mechanisms that improve its overall performance, lower the optimization time, and reduce the negative behavior which is typically connected with ACO-based algorithms (such as prematurely falling into local optima, or the impact of setting of control parameters on the convergence for different problem configurations). The most significant novel techniques, implemented for the first time to solve the multidepot vehicle routing problem (MDVRP), are as follows: 1) node clustering where transition vertices are organized into a set of candidate lists called clusters and 2) adaptive pheromone evaporation which is adapted during optimization according to the diversity of the population of ant solutions (measured by information entropy). Moreover, a new termination condition, based also on the population diversity, is formulated. The effectiveness of the proposed algorithm for the MDVRP is evaluated via a set of experiments on 23 well-known benchmark instances. Performance is compared with several state-of-the-art metaheuristic methods; the results show that the proposed algorithm outperforms these methods in most cases. Furthermore, the novel mechanisms are analyzed and discussed from points of view of performance, optimization time, and convergence. The findings achieved in this article bring new contributions to the very popular ACO-based algorithms; they can be applied to solve not only the MDVRP, but also, if adapted, to related complex NP-hard problems.**

***Index Terms*—Adaptive pheromone evaporation, ant colony optimization (ACO), entropy, multidepot vehicle routing problem, node clustering.**

I. Introduction

THE MULTIDEPOT vehicle routing problem (MDVRP) is a well-known NP-hard combinatorial optimization problem related to the famous traveling salesman problem (TSP) formulated at Princeton University in the 1930s [1]. The problem has been applied in many domains of human activities such as logistics, transportation, distribution, navigation, military, etc. [2].

Manuscript received 5 May 2022; revised 7 September 2022 and 21 November 2022; accepted 7 December 2022. Date of publication 19 December 2022; date of current version 1 December 2023. This work was supported by the Ministry of the Interior of the Czech Republic through the Project An Artificial Intelligence-Controlled Robotic System for Intelligence and Reconnaissance Operations under Grant VJ02010036. *(Corresponding author: Petr Stodola.)*

The authors are with the Department of Intelligence Support, University of Defence, 66210 Brno, Czech Republic (e-mail: petr.stodola@unob.cz; jan.nohel@unob.cz).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEVC.2022.3230042.

Digital Object Identifier 10.1109/TEVC.2022.3230042

A. Problem Formulation

The problem is formulated as follows. Let $D = \{d_1, d_2, \ldots, d_m\}$ be a set of vehicles (depots) of size m, and $C = \{c_1, c_2, \ldots, c_n\}$ be a set of customers of size n. Let G be a graph whose vertices (nodes) comprise all the customers and vehicles: $V = D \cup C = \{d_1, \ldots, d_m, c_1, \ldots, c_n\} = \{v_1, v_2, \ldots, v_{n+m}\}$. The aim of the problem is to serve all the customers by vehicles (each customer must be served just once by any vehicle) so that the total distance traveled is as small as possible.

Route R_k for vehicle $d_k \in D$ represents a sequence of customers to be served in the exact order: $R_k = (r_1^k, r_2^k, \ldots, r_{n_k}^k)$ where $r_j^k \in V$, and n_k is the number of vertices to be visited by vehicle d_k including its depot ($r_1^k = r_{n_k}^k = d_k$). The solution to the problem is a sequence of routes: $R = (R_1, R_2, \ldots, R_m)$. The objective function is in

$$|R| = \sum_{k=1}^{m} |R_k| = \sum_{k=1}^{m} \sum_{j=2}^{n_k} \left| r_j^k - r_{j-1}^k \right|. \tag{1}$$

Several constraints must be satisfied. Constraint (2) ensures that all customers must be served, constraint (3) forces each vehicle to start and end its route at the depot, and constraint (4) ensures that each customer is served just once

$$\left\{ r_1^1; \ldots; r_{n_1}^1; r_1^2; \ldots; r_{n_2}^2; r_1^m; \ldots; r_{n_m}^m \right\} = C \text{ if } r_j^k \in C \tag{2}$$

$$\left\{ r_1^1; r_{n_1}^1; r_1^2; r_{n_2}^2; r_1^m; r_{n_m}^m \right\} = D \tag{3}$$

$$r_i^k \neq r_j^k \text{ when } r_i^k, r_i^k \in C \text{ and } i \neq j. \tag{4}$$

Moreover, each customer demands a certain amount of goods [marked $r_j^{k(\text{demand})}$], and vehicles in depots have a maximum capacity [marked $d_k^{(\text{capacity})}$] which cannot be exceeded. This means that the vehicle must return back to its depot if its capacity would be exceeded by visiting the next customer on its route (thus emptying its load); after that, the vehicle can continue to serve the remaining customers on the route. Route R_k may be thus composed of more than 1 consecutive subsequences $R_k^i \subseteq C$ comprised of customers only (i.e., any part of the route without returning to the depot): $R_k = (d_k; R_k^1; d_k; R_k^2; d_k; \cdots; d_k)$. Then, capacity constraint (5) must be satisfied

$$\sum_{R_k^i} r_j^{k(\text{demand})} \leq d_k^{(\text{capacity})} \text{ for all } R_k^i \subseteq R_k. \tag{5}$$

B. Contributions

In this article, the novel metaheuristic algorithm based on the ant colony optimization (ACO) theory is proposed for the MDVRP. This algorithm implements several unique techniques which improve its performance and reduce the negative behavior typically connected with ACO-based methods (particularly the impact of the setting of control parameters on performance in connection with the graph configuration, and the tendency to fall prematurely into a local optimum).

The algorithm presented in this article extends the original ACO algorithm proposed by Stodola in [3]. Some of the key techniques implemented in the original are as follows.

1) Stochastic selection of depots before the node transitions based on the pheromone potential.
2) Hybridization with the simulated annealing principle; this is used when determining a solution from the population to update the pheromone matrix.
3) Modified k-Opt local search optimization applied not only to individual routes in the solution, but also to a pair of routes (based on a mutual exchange of nodes).

This algorithm has been extended by three novel techniques which further improve its performance. These techniques, originally developed by the authors for the TSP in [4], have been modified and adapted for the MDVRP, and integrated into the algorithm. They are as follows.

1) *Node Clustering Principle:* For every vertex, candidate lists of transition vertices (called clusters) are created. The vertices in clusters are organized based on the probability that they will be a part of the optimal solution; the higher the probability, the higher the significance of the cluster. The probability is not influenced only by the distance between vertices; the graph configuration also is considered using the sectorization principle. Node clustering does not restrict the pheromone attraction principle to be used only to a small set of vertices in a single candidate list, but enables to access a complete set of transition vertices.
2) *Adaptive Pheromone Evaporation:* The adaptive pheromone evaporation has been proposed because the right speed of evaporation is one of the most critical factors influencing the performance of the algorithm. Thus, the speed of the evaporation reflects the current optimization progress measured by the information entropy expressing the diversity of solutions in the population.
3) *Entropy-Based Termination Condition:* A new termination condition has been formulated based on the information entropy. The main idea behind this is that when the entropy is low (i.e., solutions in the population are the same or similar to one another), the probability of improving the best solution is also low and thus the optimization can be terminated.

Based on these new techniques, the proposed algorithm is called the adaptive ACO with node clustering (AACO-NC).

C. Motivation

Prakasam and Savarimuthu [5] and Han et al. [6] of this article chose the ACO-based methods because these proved to be very successful in solving discrete optimization problems. Several novel techniques have been proposed and verified. The practical use of the AACO-NC in the real software application (see details below) is based on the MDVRP transformation into various real-world problems.

The AACO-NC has been integrated into the tactical decision support system (TDSS) as a key optimization method. This system is being developed at the University of Defence, Czech Republic, in long-term research. The aim is to support military commanders of the Czech Armed Forces on the battlefield in their decision making [7]. The TDSS is composed of a set of models of military tactics. Commanders can use this system to plan their operations when one of the models is compatible with the task at hand. The system plans the variants to solve the task along with the second-order effects, and presents them to the commanders who can decide the next course of action (they can choose one of the variants which is immediately transmitted to subordinate soldiers or robotic systems for execution). More information about this topic can be found in [8], [9], [10], [11], [12], [13], [14], and [15].

Some of the models of military tactics, in which the proposed algorithm is implemented, are as follows.

1) Cooperative reconnaissance of the area of intelligence responsibility via a swarm of unmanned aerial systems (UASs). The algorithm is used to plan trajectories of individual cooperating UASs so that the reconnaissance is as fast and complete as possible.
2) Cooperative reconnaissance of the area of intelligence responsibility using a group of ground elements (soldiers or unmanned ground systems).
3) Persistent surveillance of the area or object of interest via a swarm of UASs.
4) Logistics to units on the battlefield via a group of supply vehicles.

D. Article Organization

This article is further organized as follows. Section II reviews the literature particularly with respect to the novel improvements. In Section III, the proposed algorithm is presented with a focus on the novel techniques. Section IV shows the experimental results on a set of benchmark instances, and compares the performance of the AACO-NC with other state-of-the-art methods. The analysis and discussion about the impact of the novel techniques on the performance and behavior of the algorithm are in Section V. Finally, Section VI concludes this article and suggests future work for the authors.

II. Literature Review

The MDVRP is a well-known NP-hard problem that attracts the full attention of researches. Since it was formulated in 1959 [16], many different methods have been proposed, both exact and stochastic. The most recent survey of the MDVRP [2] discusses mathematical models, state-of-the-art solution methods, and real-life applications.

This section is organized as follows. First, some examples of metaheuristic algorithms used for the MDVRP (or

related problems) are introduced in Section II-A. The bio-inspired algorithms follow in Section II-B as a strong class of stochastic methods. Then, ACO-based methods are presented in Section II-C with a focus on modern techniques and mechanisms integrated to enhance the overall performance. Finally, Sections II-D and II-E aim at the current state of research in the areas that are related to the key contributions of this article—node transition and adaptive parameters control. No part of this section can bring the complete overview of scientific works; therefore, only the state-of-the-art studies and/or key studies related in some aspect to this work are included.

A. Metaheuristic Algorithms

Metaheuristic algorithms have proved to be very successful for the constrained combinatorial problems. Genetic algorithms have been often used for the MDVRP; Singh et al. [17] proposed an improved genetic algorithm with various crossover operators and preassigning customers to their nearest depots. The genetic algorithm combined with the enhanced ordered distance vector (ODV) for population initialization is introduced in [18]. Another popular approach is the variable neighborhood search (VNS); the approach in [19] integrates biased-randomization strategies within a VNS framework in order to better guide the searching process. The variable tabu neighborhood search (VTNS) algorithm is proposed in [20]; it combines the VNS with a tabu shaking mechanism in the diversification phase of the search. The imperialist competitive algorithm (ICA) has been also often used for the combinatorial problems. Dzalbs et al. [21] proposed the ICA with independence and constrained assimilation (ICA-ICA) which combines the classical ICA assimilation and revolution operators, while maintaining population diversity.

B. Bio-Inspired Algorithms

In recent years, bio-inspired methods have become very popular and widely used for combinatorial problems. Rapanaki et al. [22] proposed a variant of the Artificial Bee Colony algorithm for the Multiobjective Energy Reduction MDVRP. The Enhanced Firefly Algorithm, which improves the solution quality by using the Clarke and Wright savings algorithm and a local search technique for the Capacitated Vehicle Routing Problem, is put forth in [23]. A hybrid nature-inspired heuristic algorithm integrating the Clarke and Wright savings algorithm, the Sweep Algorithm, and the Multiobjective Particle Swarm Optimization algorithm to solve the Multidepot Green VRP is proposed in [24].

C. ACO Algorithms With Various Improvement Techniques

ACO is one of the most popular metaheuristic optimization techniques inspired by the behavior of ants when searching for food. It has been applied to a wide range of combinatorial optimization problems. These algorithms often combine the basic ACO-based principle with particular techniques and mechanisms to enhance performance and to reduce the negative effects typically connected with the ACO algorithms. Some of the most recent research of the ACO-based algorithms applied to the MDVRP or related problems are [25] (ACO with an improved approach in updating the pheromone matrix), [26] (clustering nodes into a desired number of groups using k-means clustering algorithm, then using ACO to generate routes for each cluster), [27] (ACO with the scanning strategy and crossover operation), [28] (ACO with two distinct types of ants, the first for assigning customers to depots, the second to generate routes), [29] (ACO based on a polygonal circumcenter used for assigning customers to depots). A lot of ACO algorithms are also complemented by some local search optimization technique, mostly k-Opt local search, to enhance the overall performance, e.g., [30], [31], and [32].

D. ACO Algorithms—Node Transition Techniques

The key phase of ACO-based algorithms is the sequential nodes transition into ant solutions. In the original approach, any node that is not yet part of the solution can be selected and inserted. The probability of selecting a node depends typically on the heuristic information (reverse distance between the last node inserted and this node) and the strength of the pheromone trail. These probabilities are necessary to calculate for all the free nodes which leads to the quadratic dependence on the graph size. The idea to restrict the set of nodes for selection is therefore logical not only to reduce the computational complexity, but also to increase the overall performance and decrease the memory requirements [33]. The restricted set of nodes for selection is typically called a candidate set or a candidate list.

In the MDVRP and related graph theory problems, a single candidate list has typically a fixed size and the nodes are selected using the nearest neighbor principle (the general assumption is that local transitions lead to good solutions), i.e., the nodes closest to the last customer in the solution are inserted into the candidate list [34]. An example of this approach is in [35], where the authors developed a hybrid ACO algorithm to solve the open vehicle routing problem. In the transition phase, the customers are selected from the candidate lists comprised of the closest customers; the size of the candidate list is limited to some portion of the total number of customers. Another example is in [36]; the authors proposed the vectorized candidate set selection technique (VCSS) for a parallel ACO algorithm, where the candidate list is generated from a set of the nearest free neighbors using the roulette wheel principle. A different approach is used in [37]; instead of creating a list of candidate nodes, the pheromone matrix is restricted to the constant number of nearest neighbors for each node. All these approaches need some strategy to select a customer when there is no free node in the candidate list. The easiest way is to select the nearest free customer. However, a more advanced procedure is proposed in [37]: a pheromone value is mapped to every edge which is part of the best route; a node with the highest pheromone value assigned to the edge adjacent to the last node in the route is then inserted.

The node clustering principle introduced in this article brings novelty in two ways. First, there is not a single candidate list but a set of candidate lists comprising a complete number of vertices. The selection of a vertex for transition is performed in two phases: first, a candidate list from the set

is determined, then a vertex from this list is selected. In both phases, the probability of selecting either a list or a vertex depends on the heuristic information and the intensity of the pheromone trails. Thus, the principal strength of the ACO is preserved for the full set of vertices. Moreover, this approach does not need a special strategy if there is not a free vertex inside the single candidate list. The second enhancement is that the vertices are not organized in clusters strictly using the nearest neighbor approach, but the graph configuration is also considered using the sectorization principle.

E. ACO Algorithms—Adaptive Parameters Control

Some research has been recently conducted in the area of adaptivity of ACO-based algorithms in order to reduce negative effects, such as falling into local optima or the influence of the setting of control parameters on performance for difference graph configurations. A nucleolus game learning strategy in a set of ant colonies was proposed in [38] to improve the population diversity; the idea is to share pheromone distribution among these colonies when the algorithm stalls. Meng et al. [39] proposed an adaptive stagnation avoidance strategy in which a cooperative game model based on the population diversity determines the selection of the appropriate pheromone matrix for different colonies.

The adaptive approach to evolving control parameters has been addressed in several research articles. The alpha and beta control parameters are adjusted in [40] using a set of rules based on exploring the parameter and fitness landscape during the search. Li et al. [41] used the greedy strategy for the dynamic change of alpha and beta control parameters to accelerate the convergence speed linearly with iterations. The linear change of alpha and beta control parameters and, in addition, the evaporation coefficient is introduced in [42].

The diversity of the population of ants is measured in some studies by information entropy [39], [43]. However, to the best knowledge of the authors of this article, information entropy has not yet been used for the adaptive evolution of control parameters (particularly the evaporation coefficient) as it is used in this article. Using this principle, the convergence speed is controlled via the speed of the pheromone evaporation according to the current state of the population diversity (greater diversity enables faster convergence, and vice versa).

III. Algorithm

This section is aimed at the key principles of the proposed AACO-NC algorithm. This algorithm is inspired by the behavior of ants in nature when searching for food and it implements several techniques which improve its performance and behavior for the MDVRP.

Fig. 1 presents the key phases of the AACO-NC in pseudocode. The input of the algorithm is a set of graph vertices V (customers and depots) and control parameters which are as follows.

1) n_{ants}: Number of ants in colonies.
2) n_{freq}: Frequency of the local optimization.
3) n_{prim}: Number of primary clusters.
4) n_{size}: Number of vertices in clusters.

AACO-NC($V, n_{ants}, n_{freq}, n_{size}, n_{sect}, n_{prim}, T_{update}, \alpha_{update}, \rho_{min}, \rho_{max}, \delta, \alpha, \beta$)

1. $|R| = \infty$
2. $iter = 0$
3. Initialize pheromone matrixes τ
4. **For each** $v_i \in V$ **do**
5. $\quad K^{(v_i)} = \text{CreateClusters}(C, v_i, n_{size}, n_{sect})$
6. **While** not terminated
7. $\quad |R^{best}| = \infty$
8. $\quad iter = iter + 1$
9. $\quad$ **For** $a = 1$ to n_{ant} **do**
10. $\quad\quad R^a = \text{AntSolution}(V, K, \tau, n_{prim}, \alpha, \beta)$
11. $\quad\quad$ **If** $|R^a| < |R^{best}|$ **then do**
12. $\quad\quad\quad R^{best} = R^a$
13. $\quad$ **If** $iter \bmod n_{freq} = 0$ **then do**
14. $\quad\quad R^{best} = \text{LocalOptimization}(V, R^{best})$
15. $\quad$ **If** $|R^{best}| < |R|$ **then do**
16. $\quad\quad R = R^{best}$
17. $\quad$ Update pheromone matrixes τ
18. $\quad$ Calculate evaporation coefficient ρ
19. $\quad$ Evaporate pheromone matrixes τ using ρ
20. Return R

Fig. 1. Key phases of the AACO-NC.

5) n_{sect}: Number of sectors.
6) T_{update}: Temperature updating coefficient.
7) α_{update}: Temperature cooling coefficient.
8) ρ_{min}, ρ_{max}: Minimum and maximum limits of the pheromone evaporation coefficient.
9) δ: Pheromone updating coefficient.
10) α: Distance probability coefficient.
11) β: Pheromone probability coefficient.

The algorithm works with a number of ant colonies equal to the number of depots (one colony is located at each depot); ants starting from these colonies represent vehicles; their movements represent routes in a solution. There is the same number of ants in each colony controlled by parameter n_{ants}. A pheromone matrix is created for each colony. At the initial phase of the algorithm (point 3), the pheromone matrixes are initialized using (6). Also, clusters for every vertex of the graph are created (points 4 and 5)—see Section III-A for details

$$\tau_{ij}^{k} = 1 \text{ for all } v_i, v_j \in V \text{ and } d_k \in D. \tag{6}$$

The algorithm runs in iterations (points 6 to 19). In each iteration, solution R^a is found (point 10) for every set of ants (one ant from each colony belongs to this set) using the novel node clustering principle (see Sections III-A and III-B for details). The best solution found in an iteration is labeled as R^{best} (points 11 and 12). Then, the local optimization process may be applied to solution R^{best}; it is applied with frequency given by control parameter n_{freq} (points 13 and 14). The local optimization uses the modified k-Opt local search algorithm for individual routes in the solution, but also for the mutual exchange of vertices between any pair of routes. More details about the local optimization can be found in [3].

At the end of each iteration, pheromone matrixes are updated (point 17), and then evaporated (point 19)—see Section III-C for details. The speed of the evaporation is controlled via the pheromone evaporation coefficient ρ which is set adaptively in the range $(\rho_{\min}, \rho_{\max})$ in each iteration (point 18)—see Section III-D. The best solution found so far R is stored (points 14 and 15) in each iteration and returned (point 20) when the algorithm is terminated (see Section III-E for termination conditions).

A. Node Clustering Principle

Node clustering is the key principle improving the performance of the algorithm. This principle is used in the phase of creating a solution R^a when searching for the next node to be inserted into one of the routes of vehicles (ants).

A cluster is a set of vertices of a defined size n_{size} $(1 \leq n_{\text{size}} \leq n)$. For every vertex $v_i \in V$, a set of clusters $K^{(v_i)} = \{K_1^{(v_i)}, K_2^{(v_i)}, \ldots\}$ is created independently so that they together contain all the customers in the graph except vertex v_i itself. Vertex v_i, for which clusters $K^{(v_i)}$ are created, is then referred to as the cluster vertex for these clusters.

Customers are inserted into clusters based on the probability of being the neighboring vertices for cluster vertex v_i on the optimal route. Customers closer to the cluster vertex are more probable, so they are placed into clusters with lower indexes [the closest vertices are placed into the first cluster $K_1^{(v_i)}$].

However, based on the topology of the problem at hand, even customers relatively distant to the cluster vertex may be the neighboring vertices on the optimal route. This may occur when the problem is composed of several separate groups of customers. For this reason, a simple algorithm for creating clusters was proposed which takes this possibility into account. This algorithm can be used when the positions of vertices in their space representation are known (Euclidean or geographical).

The algorithm for creating clusters is in Fig. 2. The principle lies in dividing the space into a number of sectors n_{sect} $(1 \leq n_{\text{sect}} \leq n_{\text{size}})$. In the first phase (points 4 to 7), the closest vertex in each sector is inserted into the first cluster. In the next phase (points 8 to 14), the first cluster is completed by other vertices closest to v_i (up to n_{size} if $n_{\text{size}} > n_{\text{sec}}$), and other clusters are gradually filled by remaining vertices; the ones closer to v_i lie in clusters with lower indexes.

The first n_{prim} clusters of $K^{(v_i)}$ for each cluster vertex v_i are called primary clusters. The vertices sequentially inserted into routes of vehicles are preferentially selected from the primary clusters. See more details in the next section.

B. Ant Solution Using the Node Clustering Principle

Fig. 3 presents the algorithm for generating a solution for a set of ants using the node clustering principle. Routes R_d for individual vehicles at depots $d \in D$ are sequentially generated in a loop (points 6 to 17). Each vehicle starts from its depot (points 2 and 3). Set V_{free} (point 1) contains all the customers not visited by any vehicle so far. The last vertex (customer or depot) inserted into each route is continuously

CreateClusters$(V, v_i, n_{size}, n_{sect})$
1. $id = 1$
2. $K_{id}^{(v_i)} = \emptyset$
3. $V_{free} = V - \{v_i\}$
4. **For** $j = 1$ to n_{sect} **do**
5. | Find closest vertex $v \in V_{free}$ to v_i in sector j
6. | $K_{id}^{(v_i)} = K_{id}^{(v_i)} + \{v\}$
7. | $V_{free} = V_{free} - \{v\}$
8. **While** $V_{free} \neq \emptyset$ **do**
9. | **If** $\left|K_{id}^{(v_i)}\right| \geq n_{size}$ **then do**
10. | | $id = id + 1$
11. | | $K_{id}^{(v_i)} = \emptyset$
12. | Find closest vertex $v \in V_{free}$ to v_i
13. | $K_{id}^{(v_i)} = K_{id}^{(v_i)} + \{v\}$
14. | $V_{free} = V_{free} - \{v\}$
15. Return $K^{(v_i)}$

Fig. 2. Algorithm for creating clusters.

stored in variable v_d (points 4 and 15), and the current load of the vehicles in variable q_d (points 5 and 16).

The first step in the loop is the selection of the depot $d \in D$ (point 7); one of the vertices from set V_{free} is then inserted to the route R_d. The algorithm for the depot selection is elaborated in Fig. 4.

The next step is the selection of the cluster $K_k^{(v_d)}$ (point 8); a vertex to be inserted into the route R_d will be chosen only from free (as yet unvisited) vertices in this cluster; set $V_{\text{candidates}}$ contains those vertices (see the intersection of sets $K_k^{(v_d)}$ and V_{free} in point 9). The algorithm for the cluster selection is elaborated in Fig. 5.

The final step is the selection of the vertex $v \in V_{\text{candidates}}$ to be inserted into the route R_d (point 10). This algorithm is elaborated in Fig. 6. The selected vertex v is then inserted into the route R_d (point 14); however, the vehicle would return to its depot before visiting the vertex (thus emptying its load), if its capacity were exceeded (points 11 to 13). Then, the loop continues until all the customers are visited (i.e., V_{free} is empty). All the vehicles then return to their depots (points 18 to 19), and the ant solution R is returned (point 20).

Fig. 4 shows the algorithm for selecting a depot used in point 7 of the AACO-NC algorithm in Fig. 3. For each depot, the probability of its selection is calculated (points 1 to 7), and one of the depots is selected with this probability distribution using a simple roulette wheel principle (point 8). The probabilities are proportional to the pheromone potential of each depot; this pheromone potential is represented by the sum of pheromone trails from the last vertex v_d on the route to all available candidate nodes in the primary clusters.

When the depot $d \in D$ is determined, its route R_d will be expanded by a vertex from one of the clusters (primary clusters are preferred to other clusters). This cluster is selected using the algorithm in Fig. 5. For each cluster in primary clusters, the average heuristic information η_k (inverse distance) and average pheromone trail between the cluster node v_d and all

AntSolution$\left(V=\{D,C\},K,\tau,n_{prim},\alpha,\beta\right)$
1. $V_{free}=C$
2. **For each** $d\in D$ **do**
3. $\quad R_d=\{d\}$
4. $\quad v_d=d$
5. $\quad q_d=0$
6. **While** $V_{free}\neq\emptyset$ **do**
7. $\quad d=\text{SelectDepot}\left(v_d,V_{free},D,K^{(v_d)},\tau,n_{prim}\right)$
8. $\quad k=\text{SelectCluster}\left(d,v_d,V_{free},K^{(v_d)},\tau,n_{prim},\alpha,\beta\right)$
9. $\quad V_{candidates}=V_{free}\cap K_k^{(v_d)}$
10. $\quad v=\text{SelectCustomer}(d,v_d,V_{candidates},\tau,\alpha,\beta)$
11. $\quad$ **If** $q_d+v^{(demand)}>d^{(capacity)}$ **then do**
12. $\quad\quad R_d=R_d+\{d\}$
13. $\quad\quad q_d=0$
14. $\quad R_d=R_d+\{v\}$
15. $\quad v_d=v$
16. $\quad q_d=q_d+v^{(demand)}$
17. $\quad V_{free}=V_{free}-\{v\}$
18. **For each** $d\in D$ **do**
19. $\quad R_d=R_d+\{d\}$
20. Return $R=\left\{R_1,R_2,\ldots,R_{|D|}\right\}$

Fig. 3. Algorithm for finding an ant solution.

SelectDepot$\left(v_d,V_{free},D,K^{(v_d)},\tau,n_{prim}\right)$
1. **For each** $d_i\in D$ **do**
2. $\quad V_{cand}=\emptyset$
3. $\quad$ **For** $k=1$ to n_{prim} **do**
4. $\quad\quad V_{cand}=V_{cand}+V_{free}\cap K_k^{(v_d)}$
5. $\quad p(d_i)=\sum_{v_j\in V_{cand}}\tau_{v_dv_j}^{d_i}$
6. $p_{sum}=\sum_{d_i\in D}p(d_i)$
7. $p(d_i)=p(d_i)/p_{sum}$
8. Select $d_i\in D$ based on probabilities $p(d_i)$
9. Return d_i

Fig. 4. Algorithm for selecting a depot.

available (as yet unvisited) vertices are calculated (points 1 to 5). If there is not a single available vertex in the primary clusters (this is however a very rare situation), the first cluster behind the primary clusters with at least one available vertex is selected and returned (points 8 to 11). Otherwise, the probability of selecting individual primary clusters is proportional to the multiple of the calculated average heuristic information and the pheromone trail (points 12 to 13). The influence of the heuristic information and the pheromone trail on the probability is controlled by parameters α and β. Then, the cluster is selected based on these probability distributions using the simple roulette wheel principle (point 14).

When the cluster $K_k^{(v_d)}\in K^{(v_d)}$ is determined, one of the as yet unvisited customers in this cluster is selected using the algorithm in Fig. 6. The probability of selecting a vertex (points 1 to 2) is calculated using the standard ACO principles: it is proportional to the multiple of the heuristic information and the pheromone trail between the vertex and

SelectCluster$\left(d,v_d,V_{free},K^{(v_d)},\tau,n_{prim},\alpha,\beta\right)$
1. **For** $k=1$ to n_{prim} **do**
2. $\quad V_{cand}=V_{free}\cap K_k^{(v_d)}$
3. $\quad$ **If** $V_{cand}=\emptyset$ **then** $\eta_k=\tau_k=0$ **else do**
4. $\quad\quad \eta_k=|V_{cand}|\cdot\sum_{v_j\in V_{cand}}\left|v_d-v_j\right|^{-1}$
5. $\quad\quad \tau_k=\frac{1}{|V_{cand}|}\cdot\sum_{v_j\in V_{cand}}\tau_{v_dv_j}^{d}$
6. $\eta_{sum}=\sum_{k=1}^{n_{prim}}{\eta_k}^{\alpha}$
7. $\tau_{sum}=\sum_{k=1}^{n_{prim}}{\tau_k}^{\beta}$
8. **If** $\eta_{sum}=0$ **then do**
9. $\quad$ **For** $k=n_{prim}+1$ to $\left|K^{(v_d)}\right|$ **do**
10. $\quad\quad V_{cand}=V_{free}\cap K_k^{(v_d)}$
11. $\quad\quad$ **If** $V_{cand}\neq\emptyset$ **then** return k
12. **For** $k=1$ to n_{prim} **do**
13. $\quad p\left(K_k^{(v_d)}\right)=\frac{{\eta_k}^{\alpha}\cdot{\tau_k}^{\beta}}{\eta_{sum}\cdot\tau_{sum}}$
14. Select $K_k^{(v_d)}\in K^{(v_d)}$ based on probabilities $p\left(K_k^{(v_d)}\right)$
15. Return k

Fig. 5. Algorithm for selecting a cluster.

SelectCustomer$(d,v_d,V_{cand},\alpha,\beta,\tau)$
1. **For each** $v_i\in V_{cand}$ **do**
2. $\quad p(v_i)=\frac{|v_d-v_i|^{-\alpha}\cdot\left(\tau_{v_dv_i}^{d}\right)^{\beta}}{\sum_{v_j\in V_{cand}}|v_d-v_j|^{-\alpha}\cdot\left(\tau_{v_dv_j}^{d}\right)^{\beta}}$
3. Select $v_i\in V_{cand}$ based on probabilities $p(v_i)$
4. Return v_i

Fig. 6. Algorithm for selecting a vertex.

the cluster vertex v_d (again controlled by parameters α and β). The vertex is then selected using the simple roulette wheel principle (point 3).

C. Pheromone Matrix Update and Evaporation

The update of the pheromone matrix (see point 17 in the algorithm in Fig. 1) uses the principles of the simulated annealing optimization method. This method uses the Metropolis criterion in each iteration to decide whether to accept the newly transformed solution, or preserve the original solution. If the transformed solution is better than the original, the transformed solution is always accepted; however, even the worse solution can be accepted. This idea extends the search in the search space, thus preventing the solution from falling into some local optima.

The same idea is applied in the AACO-NC when determining which solution is used to update the pheromone matrix: 1) the best solution found in an iteration R^{best} or 2) the best solution found so far R. In general, using the former results in maintaining the diversity of the population but also in slowing the convergence down; moreover, the promising solution can be sometimes lost. Therefore, it is advantageous to use the best solution found so far for the update from time to time

instead. It results in faster convergence as well as in higher performance.

The probability of using either R^{best} or R is determined by the Metropolis criterion using (7). If R^{best} is better than R (i.e., the new best solution is found in an iteration), then R^{best} is always used. Otherwise, the R^{best} is used with the probability which depends on the percentual difference between R^{best} and R, and the current value of temperature T_{update} (the higher the temperature, the higher the probability of using the worse solution to update the pheromone matrix). The temperature, as one of the key control parameters of the AACO-NC, may change from iteration to iteration using the cooling coefficient α_{update} according to (8)

$$p\left(R^{\text{best}}\right) = 1 - p(R) = \begin{cases} e^{-\frac{\left(\left|R^{\text{best}}\right| - |R|\right)/|R|}{T_{\text{update}}}}, & \text{for } \left|R^{\text{best}}\right| > |R| \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

$$T_{\text{update}}(\text{iter}+1) = \alpha_{\text{update}} \cdot T_{\text{update}}(\text{iter}). \tag{8}$$

The solution for updating the pheromone matrix R^{update} is selected based on the calculated probabilities: $R^{\text{update}} = R^{\text{best}}$ with $p(R^{\text{best}})$, or $R^{\text{update}} = R$ with $p(R) = 1 - p(R^{\text{best}})$. The update itself is then conducted using (9); the pheromone trails lying on the routes are increased in proportion to the pheromone updating coefficient δ and the quality of the updating route (a ratio of $|R|$ to $|R^{\text{update}}|$)

$$\tau_{ij}^{k} = \tau_{ij}^{k} + x_{ij} \cdot \delta \cdot \frac{|R|}{\left|R^{\text{update}}\right|} \text{ for all } v_i, v_j \in V \text{ and } d_k \in D$$

$$x_{ij} = \begin{cases} 1, & \text{if node } v_i \text{ precedes node } v_j \text{ in } R^{\text{update}} \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

After the update, the pheromone matrix is evaporated (point 19 in the algorithm in Fig. 1) using (10). The speed of the evaporation is controlled by the pheromone evaporation coefficient ρ

$$\tau_{ij}^{k} = \tau_{ij}^{k} \cdot (1 - \rho) \text{ for all } v_i, v_j \in V \text{ and } d_k \in D. \tag{10}$$

D. Adaptive Pheromone Evaporation

One of the most critical control parameters for ACO-based algorithms is the pheromone evaporation coefficient ρ. It affects the convergence speed: higher values mean a faster convergence but also a quick fall into some local optimum which is often too far from the global optimum; lower values mean a slower convergence (or even infinite convergence). Moreover, the setting of this control parameter is problem dependent; the best value can differ significantly for different graph topologies.

To overcome this problem, the adaptive setting of the pheromone evaporation coefficient is proposed in this article. It is controlled in each iteration by the diversity of the population of ant solutions. The main idea behind this principle is that the value of the pheromone evaporation coefficient reflects the current progress in optimization. At the beginning of the optimization, the population diversity is big and therefore the convergence can be accelerated via the bigger value of the evaporation coefficient. During the optimization, when the solution converges toward some local optimum (documented by decreasing diversity in the population), the evaporation of the pheromone trails is lowered with the effect of slowing down the convergence and thus extending the search space. This principle works automatically during the whole optimization.

The population diversity is measured by the Shannon entropy using (11). The probability p_{ij} determines the probability that the edge between vertices v_i and v_j ($i \neq j$) is part of any solution in the ant population. This probability is calculated statistically based of the number of edge occurrences in solutions using (12) where n_{ij} is the number of occurrences of the edge between vertices v_i and v_j in the entire population, and $\sum n_{ij}$ is the sum of all edges used in all the solutions

$$H = -\sum_{i=2}^{n}\sum_{j=1}^{i-1} p_{ij} \cdot \log_2 p_{ij} \tag{11}$$

$$p_{ij} = \frac{n_{ij}}{\sum n_{ij}} \text{ for all } v_i, v_j \in V. \tag{12}$$

The minimum and maximum limits of entropy in the population are given by (13) and (14), respectively

$$H_{\min} = -\log_2 \frac{n_{\text{ants}}}{\sum n_{ij}} \tag{13}$$

$$H_{\max} = -\log_2 \frac{1}{\sum n_{ij}}. \tag{14}$$

The pheromone evaporation coefficient is then controlled within the predefined limits using (15) where $\rho_{\min}$ is the minimum value of the pheromone evaporation coefficient, and $\rho_{\max}$ is the maximum value. The setting of these limits is not as sensitive and dependent on the graph configuration as is the setting of a single evaporation parameter. The value adapts to the actual situation based on the current diversity in the population (linear dependence on entropy)

$$\rho = \rho_{\min} + (\rho_{\max} - \rho_{\min}) \cdot \frac{H - H_{\min}}{H_{\max} - H_{\min}}. \tag{15}$$

E. Entropy-Based Termination

The algorithm is terminated when at least one of the following conditions is satisfied.

1) Maximum number of iterations is exceeded.
2) Maximum optimization time is exceeded.
3) Maximum number of iterations without improvement in solution is exceeded.
4) Diversity of solutions in the population is too low.

The newly proposed last condition is based on the fact that when the diversity of solutions in the population is too low then the probability of finding a better solution is also low. This is because individual ant solutions are very similar to one another (there are a lot of common edges in the solutions) and, therefore, the discovery of new information is not plausible.

The diversity in the population is measured by the Shannon entropy, mentioned in the previous section. The algorithm is terminated when the relative difference between the current entropy H and the lower limit $H_{\min}$ is equal to or lower than

the relative distance expressed by coefficient ω

$$\frac{H - H_{\min}}{H_{\min}} \leq \omega. \tag{16}$$

F. Computational Complexity

The computational complexity of the original algorithm of finding a single ant solution (i.e., without the node clustering principle) is in (17); it is quadratically dependent on the number of vertices n (the quadratic dependence is caused by the calculation of transition probabilities for all the free nodes in the phase of insertion vertices into the solution). The term m represents the selection of a depot before the vertex transition

$$O(n \cdot (m + n)) = O\left(n^2 + n \cdot m\right). \tag{17}$$

The node clustering principle significantly decreases the computational complexity for this algorithm—see (18). The selection of a node to be inserted into the solution now proceeds in two phases: 1) selection of a cluster (there is n_{prim} primary clusters for selection) and 2) selection of a vertex from the cluster (there is n_{size} vertices in the selected cluster)

$$O\left(n \cdot \left(m + n_{\text{prim}} + n_{\text{size}}\right)\right). \tag{18}$$

The increase of the optimization speed is apparent as, principally, parameters n_{prim} and n_{size} are substantially lower than the total number of vertices ($n_{\text{prim}} \ll n$, $n_{\text{size}} \ll n$). Moreover, the node clustering principle preserves the key feature of the original algorithm: nodes inserted successively into the solution are always selected from the substantial set of free nodes ($n_{\text{prim}} \cdot n_{\text{size}}$) using pheromone attracting principles (it is not limited to a small set of closest vertices).

IV. Experiments and Results

This section presents the experimental results on a set of benchmark instances, and compares them with several state-of-the-art stochastic methods.

A. Benchmark Instances

As benchmarks, 23 of the well-known Cordeau's MDVRP instances [44] are used to verify the performance of the proposed AACO-NC algorithm. Table I records their dimension (number of vertices and depots) and constraints (maximum vehicle capacity and maximum route length). The layout is either random (the vertices are distributed randomly), or regular (there is at least a particular regular pattern).

TABLE I
Benchmark Instances

Instance	Vertices n	Depots m	Vehicle capacity	Route length	Layout
p01	50	4	80	∞	Random
p02	50	4	160	∞	Random
p03	75	5	140	∞	Random
p04	100	2	100	∞	Random
p05	100	2	200	∞	Random
p06	100	3	100	∞	Random
p07	100	4	100	∞	Random
p08	249	2	500	310	Random
p09	249	3	500	310	Random
p10	249	4	500	310	Random
p11	249	5	500	310	Random
p12	80	2	60	∞	Regular
p13	80	2	60	200	Regular
p14	80	2	60	180	Regular
p15	160	4	60	∞	Regular
p16	160	4	60	200	Regular
p17	160	4	60	180	Regular
p18	240	6	60	∞	Regular
p19	240	6	60	200	Regular
p20	240	6	60	180	Regular
p21	360	9	60	∞	Regular
p22	360	9	60	200	Regular
p23	360	9	60	180	Regular

B. Experimental Results

The algorithm was implemented in C++ programming language using the Visual Studio IDE. The optimizations were conducted on a personal computer with parameters as follows: Intel Core i9-10940X CPU 3.30 GHz, 32-GB RAM.

All the results were achieved using control parameters set as follows: $n_{\text{ants}} = 192$, $n_{\text{freq}} = 10$, $n_{\text{prim}} = 4$, $n_{\text{size}} = 24$, $n_{\text{sect}} = 16$, $T_{\text{update}} = 0.1$, $\alpha_{\text{update}} = 1$, $\rho_{\min} = 0.001$, $\rho_{\max} = 0.1$, $\delta = 3$, $\alpha = 1$, and $\beta = 1$. The setting of control parameters is either based on the experiments conducted in the previous research [3], [4] (parameters n_{ants}, n_{freq}, T_{update}, α_{update}, δ, α, β), or, in case of the new parameters connected with the node clustering (n_{prim}, n_{size}, n_{sect}) and adaptive pheromone evaporation ($\rho_{\min}$, $\rho_{\max}$), on the general analysis of the impact of these parameters on the behavior and performance of the algorithm. Due to the limited number of pages, there is not enough space to present the complete results here; the shortened version is presented in Section V.

For each benchmark instance, 100 optimization trials were performed. The complete results (including individual solutions for all the optimization trials) are available for download here: https://zenodo.org/record/7341076; this enables comparison of the AACO-NC with other algorithms in the future including paired statistical tests.

Table II shows the results. The best-known solutions (BKSs) were taken from [44]. These solutions are, however, outdated in some cases; better solutions for several instances can be found in the literature. However, these solutions are often only reported but not available for download (i.e., they cannot be verified). Therefore, [44] is used for the BKS in this article as it provides complete reference solutions for the benchmark instances. The BKS values in bold are proven optima [45]. Table II records the best and the worst solutions found by the AACO-NC in the optimization trials, the average values along with the standard deviation, and the optimization time in seconds.

The AACO-NC managed to find better or the same solutions for 18 instances when compared to the BKS (see the gray cells in Table II). In the remaining five instances, the difference between the best solutions and the BKS is below 1%. In four cases, the BKS were improved (p07, p09, p10, and p11). The algorithm provides very good results regardless of the graph configuration (random or regular), particularly

TABLE II
RESULTS OF THE AACO-NC

Instance	BKS	Best	Worst	Avg	Stdev	Time (s)
p01	**576.87**	576.87	605.63	586.28	8.70	8
p02	**473.53**	473.53	506.65	484.83	6.91	2
p03	**641.19**	641.19	670.30	650.31	6.49	7
p04	**1001.04**	1003.05	1032.49	1019.98	6.69	114
p05	**750.03**	750.72	780.04	758.71	6.40	27
p06	**876.50**	876.50	908.93	889.54	6.67	99
p07	885.80	884.66	916.08	897.83	6.34	68
p08	4420.95	4427.17	4553.23	4490.40	29.80	4183
p09	3900.22	3887.14	4011.69	3931.78	24.24	2358
p10	3663.02	3637.51	3781.29	3702.61	28.68	1976
p11	3554.18	3546.06	3664.88	3607.73	23.32	1483
p12	**1318.95**	1318.95	1349.00	1325.25	6.61	2
p13	**1318.95**	1318.95	1329.70	1322.43	5.00	1
p14	**1360.12**	1360.12	1422.65	1379.62	28.86	2
p15	**2505.42**	2505.42	2595.30	2552.13	19.72	48
p16	**2572.23**	2572.23	2662.73	2596.92	20.72	6
p17	**2709.09**	2709.09	2816.57	2748.97	11.58	10
p18	**3702.85**	3702.85	3823.96	3769.35	27.17	342
p19	**3827.06**	3827.06	3921.82	3858.10	20.33	34
p20	**4058.07**	4091.78	4150.92	4115.73	12.89	90
p21	5474.84	5474.84	5643.74	5514.84	37.33	3189
p22	5702.16	5702.16	5760.00	5727.51	15.15	179
p23	6095.46	6123.18	6212.81	6175.80	15.21	775
Avg	**2669.07**	**2670.04**	**2744.37**	**2700.29**	**16.30**	**652**

for larger instances: in the case of three instances with 249 nodes and random graph configuration (p09, p10, and p11), the BKS were significantly improved; in case of two instances with 360 nodes and nine depots and regular graph configuration (p21 and p22), the same solutions as the BKS were achieved.

The last row in Table II shows the average values in individual columns over all the instances. The difference between the best solutions found and the BKS is minimal (0.01%). The difference between the average and worst solutions and the BKS is 1.27% and 3.30%, respectively.

The optimization time differs sometimes substantially. It is caused, for one, by the graph configurations which may influence the convergence speed radically. The local optimization process also has a big impact on the optimization speed, especially the mutual exchange of vertices between vehicle routes (see [3] for details). This can be accelerated when the maximum route length is limited (the exchange of vertices between routes too far from one another is pointless). The effect of this acceleration can be seen on instances p21, p22, and p23: instance p21 does not have a limited maximum route length, and thus the optimization time is considerably longer than in the case of instances p22 and p23, even though all these instances have the same graph complexity (360 vertices and nine depots).

C. Comparison With Other Methods

The performance of the AACO-NC on the benchmark instances was compared to five state-of-the-art stochastic methods. These methods are comprised of three recent ACO-based algorithms as well as two different metaheuristic principles.

When choosing methods for comparison, those based on ACO principles were preferred. They were supplemented by two methods based on different metaheuristic principles. All selected methods meet the following conditions: 1) they are state-of-the-art methods with high performance; 2) they have been published in renowned scientific journals in recent years; and 3) sufficiently detailed results for benchmark instances are available in the publications. The latter condition results from the unavailability of source codes or the difficulty of making them operational; in some cases, however, the published results are not complete.

The methods for comparison are as follows.

1) *IACO:* Improved ACO with a scanning strategy [27].
2) *TSACS:* Two-stage ant colony system with the two types of ants (for assigning customers to depots and for generating routes) [28].
3) *BPC-HACO:* Hybrid ACO based on a polygonal circumcenter [29].
4) *ICA-ICA:* Imperialist competitive algorithm with independence and constrained assimilation [21].
5) *VND-TSH:* The hybrid algorithm combining Variable Neighborhood Descent and Tabu Search Heuristic [46].

Table III compares the best results found. The best values for each instance are emphasized by a gray background. The AACO-NC provided better or the same solutions for 17 instances. The last row of Table III records the average solutions over all the instances. The AACO-NC outperformed the rival algorithms in this regard followed by the ICA-ICA (gap 0.23%), IACO (gap 0.4%), BPC-HACO (gap 0.48%), TSACS (gap 0.7%), and VND-TSH (gap 0.84%). The gap is calculated using (19) where *A*vg is the average of the best solutions from Table III found by one of the algorithms (see the last row of Table III)

$$\text{gap} = \frac{\text{Avg}^{\text{Algorithm}} - \text{Avg}^{\text{AACO-NC}}}{\text{Avg}^{\text{AACO-NC}}} \cdot 100\ [\%]. \qquad (19)$$

Table IV compares the average solutions (results for the TSACS and VND-TSH are missing as they are not available in the literature). The AACO-NC, BPC-HACO, and ICA-ICA provide similar results overall: the AACO-NC is followed by BPC-HACO (gap 0.21%), ICA-ICA (gap 0.48%), and finally IACO (gap 3.44%). The gap is calculated using (19) where *Avg* is the average of the solutions from Table IV found by one of the algorithms (see the last row of Table IV). The BPC-HACO, although it has slightly lower overall performance than AACO-NC, provides better average solutions for more instances (eight cases) than the AACO-NC (six cases). Despite this, the AACO has more cases of best solutions found (17 cases) than the BCP-ACO (11 cases). This corresponds to the AACO-NC having a larger standard deviation (averaged 16.3) than the BCP-ACO (averaged 9.06).

Table V presents optimization times (in seconds) of the algorithms taken from the literature (they are not available for the TSACS and BPC-HACO). This is for purposes of illustration only because the optimizations were taken on different hardware configurations and, therefore, the deeper analysis would not be appropriate. However, some basic conclusions can be drawn. The AACO-NC is noticeably faster than other

TABLE III
COMPARISON OF THE AACO-NC WITH OTHER METHODS (BEST SOLUTIONS FOUND)

Instance	AACO-NC	IACO	TSACS	BPC-HACO	ICA-ICA	VND-TSH
p01	576.87	576.87	576.87	576.87	576.87	576.87
p02	473.53	473.53	473.53	473.53	473.53	473.53
p03	641.19	641.19	641.19	641.19	641.19	641.19
p04	1003.05	1001.59	1010.57	1003.73	1006.66	1008.62
p05	750.72	750.26	751.15	750.03	753.40	752.04
p06	876.50	876.50	880.57	876.50	876.50	882.71
p07	884.66	885.69	881.97	884.66	895.53	896.01
p08	4427.17	4482.44	4516.75	4406.68	4420.94	4417.34
p09	3887.14	3912.23	3939.52	3897.60	3900.22	3940.56
p10	3637.51	3663.00	3724.93	3663.02	3666.35	3696.31
p11	3546.06	3648.95	3624.67	3580.80	3554.18	3578.14
p12	1318.95	1318.95	1318.95	1318.95	1318.95	1318.95
p13	1318.95	1318.95	1318.95	1318.95	1318.95	1318.95
p14	1360.12	1365.68	1360.12	1360.12	1365.68	1360.12
p15	2505.42	2505.42	2505.42	2505.42	2565.67	2538.79
p16	2572.23	2587.87	2572.23	2572.23	2572.23	2572.23
p17	2709.09	2709.09	2709.09	2731.37	2709.09	2731.37
p18	3702.85	3781.04	3749.34	3741.99	3710.49	3798.58
p19	3827.06	3827.06	3827.06	3863.90	3827.06	3827.06
p20	4091.78	4058.07	4058.07	4097.06	4058.07	4097.06
p21	5474.84	5474.84	5619.95	5575.79	5495.54	5643.55
p22	5702.16	5702.16	5702.16	5718.00	5702.16	5708.36
p23	6123.18	6095.46	6078.75	6145.58	6145.58	6145.58
Avg	**2670.04**	**2680.73**	**2688.77**	**2682.78**	**2676.30**	**2692.34**

TABLE IV
COMPARISON OF THE AACO-NC WITH OTHER METHODS (AVERAGE SOLUTIONS)

Instance	AACO-NC	IACO	TSACS	BPC-HACO	ICA-ICA	VND-TSH
p01	586.28	576.87	NA	582.42	576.87	NA
p02	484.83	480.14	NA	482.36	481.24	NA
p03	650.31	646.52	NA	652.50	655.29	NA
p04	1019.98	1010.60	NA	1010.91	1015.11	NA
p05	758.71	763.84	NA	760.06	789.15	NA
p06	889.54	892.46	NA	885.24	887.71	NA
p07	897.83	886.31	NA	894.31	916.79	NA
p08	4490.40	4594.73	NA	4426.52	4493.66	NA
p09	3931.78	4105.22	NA	3908.78	3975.29	NA
p10	3702.61	3732.06	NA	3672.58	3696.71	NA
p11	3607.73	3816.79	NA	3590.26	3604.88	NA
p12	1325.25	1330.31	NA	1328.35	1359.49	NA
p13	1322.43	1343.73	NA	1325.87	1320.79	NA
p14	1379.62	1376.24	NA	1370.69	1394.01	NA
p15	2552.13	2564.32	NA	2513.26	2644.14	NA
p16	2596.92	2598.53	NA	2591.40	2577.66	NA
p17	2748.97	2746.41	NA	2750.54	2742.93	NA
p18	3769.35	3968.06	NA	3751.04	3756.70	NA
p19	3858.10	3994.65	NA	3901.37	3857.36	NA
p20	4115.73	4356.70	NA	4139.11	4134.88	NA
p21	5514.84	5889.46	NA	5596.32	5564.61	NA
p22	5727.51	6196.03	NA	5818.34	5753.71	NA
p23	6175.80	6376.24	NA	6283.17	6205.46	NA
Avg	**2700.29**	**2793.31**	**NA**	**2705.89**	**2713.24**	**NA**

methods for simpler instances with a random distribution of nodes (p01, p02, and p03); times are comparable to other algorithms for instances up to 100 nodes (p04, p05, p06, and p07). Even bigger differences are apparent for instances

TABLE V
COMPARISON OF THE AACO-NC WITH OTHER METHODS (OPTIMIZATION TIME IN SECONDS)

Instance	AACO-NC	IACO	TSACS	BPC-HACO	ICA-ICA	VND-TSH
p01	8	58	NA	NA	252	25
p02	2	61	NA	NA	372	45
p03	7	62	NA	NA	474	16
p04	114	92	NA	NA	744	119
p05	27	88	NA	NA	1218	111
p06	99	94	NA	NA	882	128
p07	68	93	NA	NA	690	96
p08	4183	146	NA	NA	3912	634
p09	2358	135	NA	NA	4056	752
p10	1976	137	NA	NA	4932	740
p11	1483	127	NA	NA	4260	506
p12	2	70	NA	NA	600	15
p13	1	74	NA	NA	534	11
p14	2	112	NA	NA	402	24
p15	48	178	NA	NA	1530	288
p16	6	192	NA	NA	960	126
p17	10	168	NA	NA	738	95
p18	342	206	NA	NA	4392	195
p19	34	213	NA	NA	2538	66
p20	90	306	NA	NA	4416	121
p21	3189	403	NA	NA	4914	1900
p22	179	419	NA	NA	5160	584
p23	775	602	NA	NA	5022	100
Avg	**652**	**175**	**NA**	**NA**	**2304**	**291**

with regular distribution of nodes (p12, p13, p14, p15, p15, and p17); this also applies for some more complex instances (p19, p20, and p22). On the other hand, the AACO-NC is visibly slower for complex instances with a random distribution of nodes (p08, p09, p10, and p11)—except for the ICA-ICA which is slower in all cases.

Wilcoxon signed-rank tests are performed to further compare the AACO-NC with the IACO, BPC-HACO, and ICA-ICA (insufficient data is available for the TSACS and VND-TSH to perform these tests). The Wilcoxon tests were chosen because the AACO-NC results are significantly different from a normal distribution (confirmed by Shapiro-Wilk's tests). The hypotheses are as follows (H_{method} denotes H_{IACO}, $H_{\text{BCP-HACO}}$, or $H_{\text{ICA-ICA}}$ according to the method benchmarked with the AACO-NC in the pairwise tests).

1) *Null Hypothesis* H_0: $\mu_{\text{AACO-NC}} = \mu_{\text{method}}$; there is no significant difference between the AACO-NC and the benchmarked method.
2) *Alternative Hypothesis* $H_{AACO-NC}$: $\mu_{\text{AACO-NC}} < \mu_{\text{method}}$; there is a significant difference between the AACO-NC and the benchmarked method (the AACO-NC provides better results).
3) *Alternative Hypothesis* H_{method}: $\mu_{\text{AACO-NC}} > \mu_{\text{method}}$; there is a significant difference between the AACO-NC and the benchmarked method: (the rival method provides better results).

Table VI presents the results of the tests. To fail to reject the null hypothesis, test statistic Z must lie within the critical interval $(-1.96; 1.96)$ for the level of significance $\alpha = 0.05$. When Z is below the lower value, $H_{\text{AACO-NC}}$ is accepted (the AACO-NC outperforms the benchmarked rival method),

TABLE VI
WILCOXON SIGNED-RANKED TESTS FOR BENCHMARKS

	AACO-NC vs IACO		AACO-NC vs BPC-HACO		AACO-NC vs ICA-ICA	
Instance	***Z***	**Hyp.**	***Z***	**Hyp.**	***Z***	**Hyp.**
p01	7.770	H_{IACO}	3.153	$H_{\text{BPC-HACO}}$	7.770	$H_{\text{ICA-ICA}}$
p02	5.766	H_{IACO}	2.995	$H_{\text{BPC-HACO}}$	4.394	$H_{\text{ICA-ICA}}$
p03	5.096	H_{IACO}	-3.397	$H_{\text{AACO-NC}}$	-6.124	$H_{\text{AACO-NC}}$
p04	8.121	H_{IACO}	8.032	$H_{\text{BPC-HACO}}$	5.897	$H_{\text{ICA-ICA}}$
p05	-6.254	$H_{\text{AACO-NC}}$	-2.507	$H_{\text{AACO-NC}}$	-8.682	$H_{\text{AACO-NC}}$
p06	-4.100	$H_{\text{AACO-NC}}$	5.329	$H_{\text{BPC-HACO}}$	1.939	H_0
p07	8.668	H_{IACO}	4.731	$H_{\text{BPC-HACO}}$	-8.682	$H_{\text{AACO-NC}}$
p08	-8.682	$H_{\text{AACO-NC}}$	8.682	$H_{\text{BPC-HACO}}$	-0.774	H_0
p09	-8.682	$H_{\text{AACO-NC}}$	7.520	$H_{\text{BPC-HACO}}$	-8.438	$H_{\text{AACO-NC}}$
p10	-7.234	$H_{\text{AACO-NC}}$	7.643	$H_{\text{BPC-HACO}}$	1.564	H_0
p11	-8.682	$H_{\text{AACO-NC}}$	6.137	$H_{\text{BPC-HACO}}$	1.368	H_0
p12	-6.065	$H_{\text{AACO-NC}}$	-5.261	$H_{\text{AACO-NC}}$	-8.682	$H_{\text{AACO-NC}}$
p13	-8.682	$H_{\text{AACO-NC}}$	-6.866	$H_{\text{AACO-NC}}$	0.615	H_0
p14	0.378	H_0	0.378	H_0	-6.763	$H_{\text{AACO-NC}}$
p15	-5.446	$H_{\text{AACO-NC}}$	8.558	$H_{\text{BPC-HACO}}$	-8.682	$H_{\text{AACO-NC}}$
p16	-1.812	H_0	2.276	$H_{\text{BPC-HACO}}$	7.812	$H_{\text{ICA-ICA}}$
p17	2.139	H_{IACO}	-2.960	$H_{\text{AACO-NC}}$	5.735	$H_{\text{ICA-ICA}}$
p18	-8.682	$H_{\text{AACO-NC}}$	5.539	$H_{\text{BPC-HACO}}$	4.246	$H_{\text{ICA-ICA}}$
p19	-8.682	$H_{\text{AACO-NC}}$	-8.620	$H_{\text{AACO-NC}}$	-0.058	H_0
p20	-8.682	$H_{\text{AACO-NC}}$	-8.431	$H_{\text{AACO-NC}}$	-8.221	$H_{\text{AACO-NC}}$
p21	-8.682	$H_{\text{AACO-NC}}$	-8.510	$H_{\text{AACO-NC}}$	-7.533	$H_{\text{AACO-NC}}$
p22	-8.682	$H_{\text{AACO-NC}}$	-8.682	$H_{\text{AACO-NC}}$	-8.572	$H_{\text{AACO-NC}}$
p23	-8.682	$H_{\text{AACO-NC}}$	-8.682	$H_{\text{AACO-NC}}$	-8.665	$H_{\text{AACO-NC}}$

when it is above the higher value, H_{method} is accepted (the benchmarked method outperforms the AACO-NC).

Wilcoxon tests show that the AACO-NC outperforms the IACO and ICA-ICA in most cases (hypothesis $H_{\text{AACO-NC}}$ accepted in 15 and 11 cases, respectively, typically for more complex problems), while these methods outperform the AACO-NC in six cases (hypotheses H_{IACO} and $H_{\text{ICA-ICA}}$ are mostly accepted for simpler problems). The BPC-HACO and the AACO-NC are comparable; hypothesis $H_{\text{AACO-NC}}$ is accepted in ten cases, $H_{\text{BPC-HACO}}$ in 12 cases, and there is not enough evidence to reject the null hypothesis in one case (p14). The BPC-HACO has higher performance mostly on instances with a random distribution of nodes, while the AACO-NC mostly on instances with regular distribution of nodes. Although the BPC-HACO provides higher performance in more cases (12 compared to 10), the higher standard deviation of the AACO-NC ensures the noticeably more best solutions found in the set of performed experiments (11 compared to 2).

V. ANALYSIS AND DISCUSSION

This section analyses the behavior of the proposed algorithm from the new enhancements point of view.

A. Node Clustering

Fig. 7 shows the impact of node clustering on the performance of the AACO-NC on eight selected benchmark instances. This selection was based of the problem complexity (from simple to the most complex) and distribution of nodes (random or regular). One representative is included in each category ($n = 50,\ 75,\ 100,\ 249$ for random distribution, and

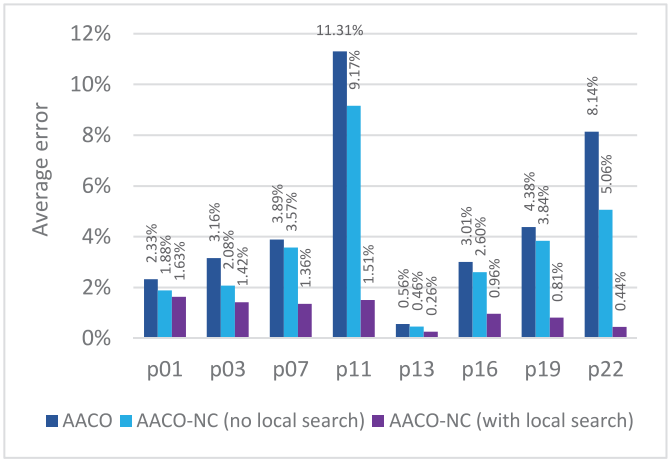

Fig. 7. Impact of node clustering on performance.

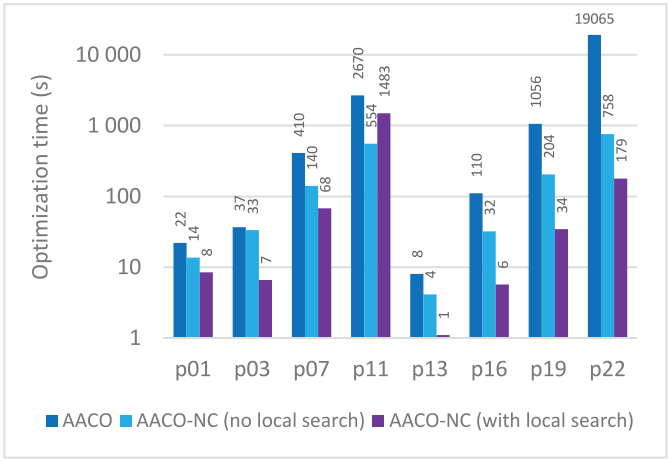

Fig. 8. Impact of node clustering on the optimization time.

$n = 80,\ 160,\ 240,\ 360$ for regular distribution). The dark blue color shows the average error (the relative difference between the average solution and the BKS) of the algorithm without using node clustering (named AACO), i.e., the full set of nodes is available for selection in the transition phase. The light blue color shows the average error of the AACO-NC, i.e., with node clustering. In both cases, the local search optimization was switched off to emphasize the influence of the node clustering principle. However, the adaptive pheromone evaporation was preserved. In all cases, performance improved; the more complex the problem, the bigger the improvement. For example, in instance p22 (360 nodes and 9 depots), the error is over 8% when not using node clustering, and it improved to about 5% when using this mechanism. Moreover, for illustration, the violet color shows the average error when the complete AACO-NC with the local search optimization is used (see results in Table II). The improvement is substantial both for random and regular graphs.

An even bigger enhancement is achieved in the optimization time—see Fig. 8. Note that the axis scale is logarithmic because of the significant differences over instances. The reduction of the optimization time depends on the problem complexity; e.g., for instance p11 (249 nodes and 5 depots),

the AACO-NC without the local search (light blue color) is almost 5 times faster compared to AACO (dark blue color), for instance p22 (360 nodes and 9 depots), it is more than 25 times faster. The local search (violet color) has a very interesting impact. In most cases (the exception is instance p11), it accelerates the optimization; it is especially extensive in the case of regular graphs (p13, p16, p19, and p22). This is caused due to a much faster convergence (the local search is very efficient for the regular graphs).

The following assumptions can be made from the analysis conducted.

1) The node clustering increases the performance of the algorithm.
2) The more complex problem, the higher the performance improvement (the performance of the general ACO algorithm without other supporting techniques deteriorates with increasing problem complexity; the node clustering reduces this deterioration).
3) It is important to integrate the general ACO algorithm with the local search optimization to further reduce the deterioration.
4) The optimization time is reduced significantly when using the node clustering.
5) The more complex problem, the higher the time acceleration [due to the significantly lower computational complexity—see (18)].
6) The local search optimization usually shorten the optimization time even though it is computationally demanding (because it accelerates convergence).

B. Adaptive Pheromone Evaporation

The impact of the pheromone coefficient on the convergence speed is shown in Fig. 9. It presents the dependence of the average error on the number of performed iterations for instance p10. This instance was chosen as one of the most complex representatives of problems with a random distribution of nodes. Two of the curves (green and blue color) indicate the optimization progress when the constant value of the pheromone evaporation coefficient was used ($\rho = 0.001$ and $\rho = 0.1$). On the other hand, the violet curve is the optimization progress with the adaptive pheromone evaporation ($\rho_{\min} = 0.001$, $\rho_{\max} = 0.1$). Adaptive evaporation ensures almost the same performance at the end of the optimization (average error 1.08%) compared to the constant evaporation when $\rho = 0.001$ (average error 0.89%). However, the convergence was much faster (with the adaptive approach, the 2% error is provided about iteration 10 000; without this approach, it takes almost twice as long to achieve the same error). When the larger value of the constant pheromone evaporation coefficient is used ($\rho = 0.1$), the convergence is fast, but at the expense of the overall performance (average error 1.98%).

Fig. 10 shows the progress of the diversity of the population of ants for the same optimization as in Fig. 9. The information entropy curve corresponds to the average error curve in Fig. 9. When the large value of the pheromone evaporation coefficient is used ($\rho = 0.1$), the diversity quickly becomes very low (blue color). On the other hand, when $\rho = 0.001$, high diversity is maintained significantly longer (green color). Adaptive evaporation ensures that the population diversity is between these two extremes (violet color); this enables to high-quality results to be achieved faster.

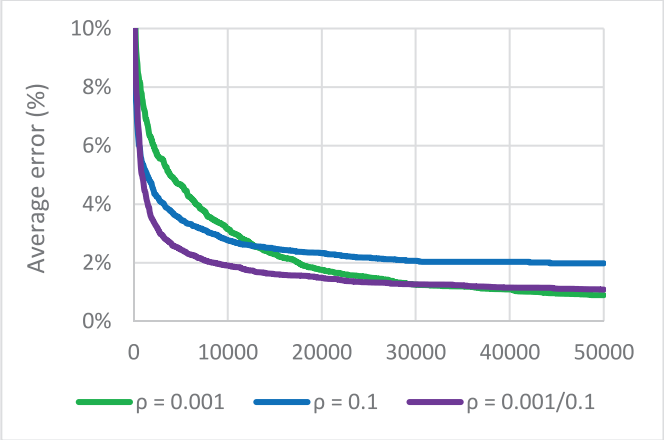

Fig. 9. Optimization progress for instance p10 using constant and adaptive pheromone evaporation.

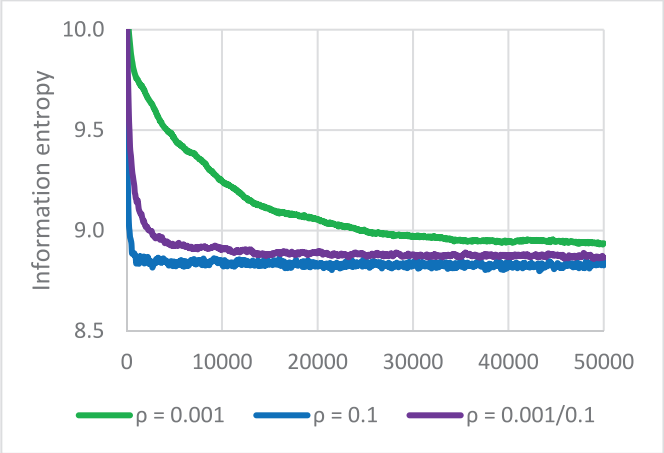

Fig. 10. Information entropy progress for instance p10 using constant and adaptive pheromone evaporation.

The same analysis is shown in Figs. 11 and 12, this time for instance p22 as a representative for the most complex instances with regular distribution of nodes. The faster convergence, than as in case of random distribution of nodes, is noticeable in Fig. 11; the best solution is usually found before iteration 10,000 when using the adaptive evaporation. Otherwise, the optimization progress (Fig. 11) as well as entropy progress (Fig. 12) have the same development as in case of the problem with a random distribution of nodes (Figs. 8 and 9).

Several assumptions about differences between algorithms with the adaptive and constant evaporation (AACO versus ACO) can be formulated.

1) AACO ensures faster convergence to a local or the global optimum than ACO when $\rho_{\min} = \rho$ and $\rho_{\max} > \rho$.
2) The overall performance of AACO is very close to the performance of ACO when $\rho_{\min} = \rho$.

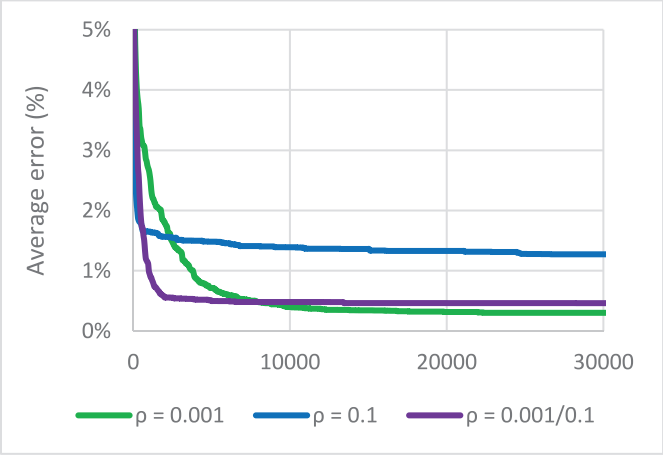

Fig. 11. Optimization progress for instance p22 using constant and adaptive pheromone evaporation.

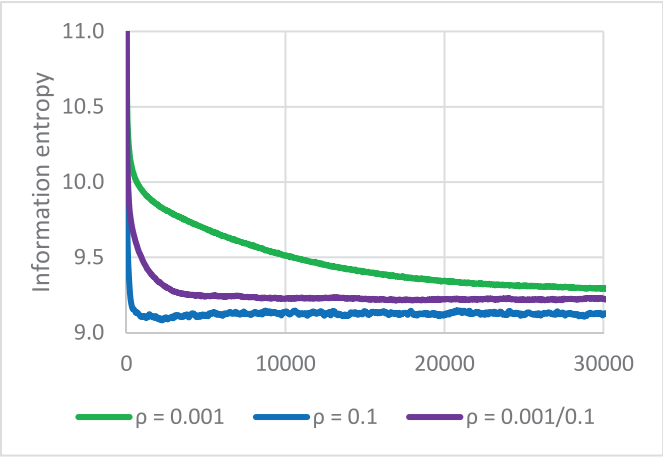

Fig. 12. Information entropy progress for instance p22 using constant and adaptive pheromone evaporation.

3) Achieving the comparable performance as with ACO, however faster, could be done via AACO when $\rho_{\min} < \rho$.
4) Solutions of similar quality are found noticeably faster via AACO when $\rho_{\min} = \rho$.

C. Entropy-Based Termination Condition

The impact of the new condition on the termination of the algorithm is shown in this section. The optimizations were executed with other termination conditions switched off, i.e., only the entropy-based condition terminated the algorithm ($\omega = 0.1$). The total number of iterations performed as well as the iteration in which the best solution was found (both values averaged over 30 experiments for each instance) were recorded and the ratio between these two values is calculated and presented in Fig. 13. This ratio represents the "wasted" optimization time, i.e., the iterations that were performed at the end of the optimization but did not improve the solution.

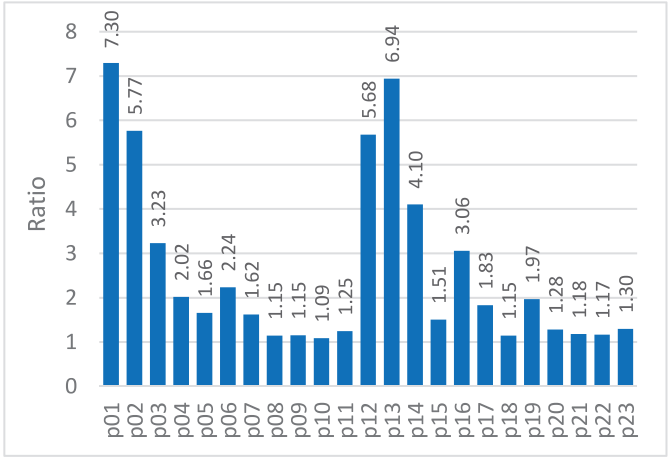

Fig. 13. Ratio of the total number of performed iterations to the iteration with the last improvement in solution.

The results show that for the simple problems ($n < 100$), the ratio is relatively high. The reason for this is that a high-quality local or the global optimum is found quickly, long before the diversity in the population decreases to the termination threshold. However, the ratio for more complex problems ($n > 100$) is mostly below 2 (or slightly exceeds 2 in two cases); the only exception to this is instance p16 (ratio above 3). For the more complex problems ($n = 249$ for random topology of nodes and $n = 360$ for regular topology of nodes), the ratio is even lower (below 1.3, that is no more than 30% of iterations are wasted). Also, the termination condition works without the significant difference both for random and regular distribution of nodes. For the simpler problems, the entropy-based termination condition is recommended to complement with another termination condition, preferably by limiting the number of iterations without improving the solution.

VI. Conclusion

The proposed algorithm with the novel techniques (node clustering, adaptive pheromone evaporation, and entropy-based termination condition) improves the overall performance when applied to solve the MDVRP. This was verified on a set of benchmark instances of various complexity, both with random and regular distribution of vertices. The comparison with other state-of-the-art metaheuristic methods showed the high potential and competitiveness of the proposed algorithm for complex combinatorial optimization problems.

The novelty in this article lies in the adaptation of the three supporting optimization mechanics to the MDVRP, and their integration into the ACO-based algorithm. Moreover, the findings achieved in this study could inspire further modifications of the proposed techniques for other combinatorial problems.

The future work of the authors will be focused on the organization of vertices into clusters. The goal is to increase the probability that the vertex forming the optimal route is included in the primary clusters; at the same time, the size

of the primary clusters should be as small as possible. The authors plan to achieve this goal via data mining techniques (the vertices will be classified into clusters by selected data mining methods).

REFERENCES

[1] M. M. Flood, "The traveling-salesman problem," *Oper. Res.*, vol. 4, no. 1, pp. 61–75, Feb. 1956, doi: 10.1287/opre.4.1.61.

[2] D. G. N. D. Jayarathna, G. H. J. Lanel, and Z. A. M. S. Juman, "Survey on ten years of multi-depot vehicle routing problems: Mathematical models, solution methods and real-life applications," *Sustain. Develop. Res.*, vol. 3, no. 1, pp. 36–47, Feb. 2021, doi: 10.30560/sdr.v3n1p36.

[3] P. Stodola, "Hybrid ant colony optimization algorithm applied to the multi-depot vehicle routing problem," *Nat. Comput.*, vol. 19, no. 2, pp. 463–475, Jun. 2020, doi: 10.1007/s11047-020-09783-6.

[4] P. Stodola, P. Otřísal, and K. Hasilová, "Adaptive ant colony optimization with node clustering applied to the travelling salesman problem," *Swarm Evol. Comput.*, vol. 70, Apr. 2022, Art. no. 101056, doi: 10.1016/j.swevo.2022.101056.

[5] A. Prakasam and N. Savarimuthu, "Metaheuristic algorithms and probabilistic behaviour: A comprehensive analysis of ant colony optimization and its variants," *Artif. Intell. Rev.*, vol. 45, no. 1, pp. 97–130, 2016, doi: 10.1007/s10462-015-9441-y.

[6] Z. Han, Y. Wang, and D. Tian, "Ant colony optimization for assembly sequence planning based on parameters optimization," *Front. Mech. Eng.*, vol. 16, no. 2, pp. 393–409, 2021, doi: 10.1007/s11465-020-0613-3.

[7] P. Stodola and J. Mazal, "Tactical decision support system to aid commanders in their decision-making," in *Modelling and Simulation for Autonomous Systems*, vol. 9991. Cham, Switzerland: Springer, 2016, pp. 396–406, doi: 10.1007/978-3-319-47605-6_32.

[8] M. Rybansky, "Determination the ability of military vehicles to override vegetation," *J. Terramech.*, vol. 91, pp. 129–138, Oct. 2020, doi: 10.1016/j.jterra.2020.06.004.

[9] J. Rada, M. Rybansky, and F. Dohnal, "The impact of the accuracy of terrain surface data on the navigation of off-road vehicles," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 3, p. 106, Feb. 2021, doi: 10.3390/ijgi10030106.

[10] D. Kristalova et al., "Modelling and simulation of microrelief impact on ground path extension," in *Modelling and Simulation for Autonomous Systems*, vol. 13207. Cham, Switzerland: Springer, 2022, pp. 93–112. doi: 10.1007/978-3-030-98260-7_6.

[11] J. Hrdina, P. Vašík, J. Procházka, L. Kutěj, and R. Ščurek, "The weighted core of games based on tactical decisions," in *Modelling and Simulation for Autonomous Systems*. Cham, Switzerland: Springer, 2020, pp. 244–252, doi: 10.1007/978-3-030-43890-6_19.

[12] Š. Hošková-Mayerová, V. Talhofer, P. Otřísal, and M. Rybanský, "Influence of weights of geographical factors on the results of multicriteria analysis in solving spatial analyses," *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 8, p. 489, Aug. 2020, doi: 10.3390/ijgi9080489.

[13] Dringuš, L. Madarász, M. Oravec, and V. Gašpar, "Concept of situational control in road tunnels," in *Proc. IEEE Int. Symp. Logist. Ind. Informat.*, Sep. 2012, pp. 113–120, doi: 10.1109/LINDI.2012.6319472.

[14] P. Stodola, J. Drozd, K. Šilinger, J. Hodický, and D. Procházka, "Collective perception using UAVs: Autonomous aerial reconnaissance in a complex urban environment," *Sensors*, vol. 20, no. 10, p. 2926, Jan. 2020, doi: 10.3390/s20102926.

[15] J. Drozd, P. Stodola, D. Křištálová, and J. Kozubek, "Experiments with the UAS reconnaissance model in the real environment," in *Modelling and Simulation for Autonomous Systems*. Cham, Switzerland: Springer, 2017, pp. 340–349, doi: 10.1007/978-3-319-76072-8_24.

[16] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manag. Sci.*, vol. 6, no. 1, pp. 80–91, Oct. 1959, doi: 10.1287/mnsc.6.1.80.

[17] V. Singh, L. Ganapathy, and A. K. Pundir, "An improved genetic algorithm for solving multi depot vehicle routing problems," *Int. J. Inf. Syst. Supply Chain Manag.*, vol. 12, no. 4, pp. 1–26, 2019, doi: 10.4018/IJISSCM.2019100101.

[18] U. Prabu, P. Ravisasthiri, R. Sriram, N. Malarvizhi, and J. Amudhavel, "EODVGA: An enhanced ODV based genetic algorithm for multi-depot vehicle routing problem," *EAI Endorsed Trans. Scalable Inf. Syst.*, vol. 6, no. 21, p. e8, Jun. 2019, doi: 10.4108/eai.10-6-2019.159099.

[19] L. Reyes-Rubiano, L. Calvet, A. A. Juan, J. Faulin, and L. Bové, "A biased-randomized variable neighborhood search for sustainable multi-depot vehicle routing problems," *J. Heuristics*, vol. 26, no. 3, pp. 401–422, 2020, doi: 10.1007/s10732-018-9366-0.

[20] M. E. Hesam Sadati, B. Çatay, and D. Aksen, "An efficient variable neighborhood search with tabu shaking for a class of multi-depot vehicle routing problems," *Comput. Oper. Res.*, vol. 133, Sep. 2021, Art. no. 105269, doi: 10.1016/j.cor.2021.105269.

[21] I. Dzalbs, T. Kalganova, and I. Dear, "Imperialist competitive algorithm with independence and constrained assimilation," in *Proc. Int. Congr. Human-Comput. Interact. Optim. Robot. Appl. (HORA)*, Jun. 2020, pp. 1–11, doi: 10.1109/HORA49412.2020.9152916.

[22] E. Rapanaki, I.-D. Psychas, M. Marinaki, and Y. Marinakis, "An artificial bee colony algorithm for the multiobjective energy reduction multi-depot vehicle routing problem," in *Learning and Intelligent Optimization*, Cham, Switzerland: Springer, 2020, pp. 208–223, doi: 10.1007/978-3-030-38629-0_17.

[23] R. Yesodha and T. Amudha, "An improved firefly algorithm for capacitated vehicle routing optimization," in *Proc. Amity Int. Conf. Artif. Intell. (AICAI)*, Feb. 2019, pp. 163–169, doi: 10.1109/AICAI.2019.8701269.

[24] Y. Wang, K. Assogba, J. Fan, M. Xu, Y. Liu, and H. Wang, "Multi-depot green vehicle routing problem with shared transportation resource: Integration of time-dependent speed and piecewise penalty cost," *J. Clean. Prod.*, vol. 232, pp. 12–29, Sep. 2019, doi: 10.1016/j.jclepro.2019.05.344.

[25] Y. Li, H. Soleimani, and M. Zohal, "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives," *J. Clean. Prod.*, vol. 227, pp. 1161–1172, Aug. 2019, doi: 10.1016/j.jclepro.2019.03.185.

[26] S. Rajak, P. Parthiban, and R. Dhanalakshmi, "Multi-depot vehicle routing problem based on customer satisfaction," *Int. J. Services Technol. Manag.*, vol. 26, nos. 2–3, pp. 252–265, Jan. 2020, doi: 10.1504/IJSTM.2020.106693.

[27] B. Yao, C. Chen, X. Song, and X. Yang, "Fresh seafood delivery routing problem using an improved ant colony optimization," *Ann. Oper. Res.*, vol. 273, no. 1, pp. 163–186, 2019, doi: 10.1007/s10479-017-2531-2.

[28] W. Zhang, Y. Gajpal, S. S. Appadoo, and Q. Wei, "Multi-depot green vehicle routing problem to minimize carbon emissions," *Sustainability*, vol. 12, no. 8, p. 3500, Jan. 2020, doi: 10.3390/su12083500.

[29] F. Wan, H. Gou, W. Pan, J. Hou, and S. Chen, "A mathematical method for solving multi-depot vehicle routing problem," *Heliyon*, early access, Oct. 2021, doi: 10.2139/ssrn.3943419.

[30] Thammano and Y. Oonsrikaw, "Improved ant colony optimization with local search for traveling salesman problem," in *Proc. IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distrib. Comput. (SNPD)*, Jul. 2019, pp. 22–27, doi: 10.1109/SNPD.2019.8935817.

[31] M. Mavrovouniotis, I. S. Bonilha, F. M. Müller, G. Ellinas, and M. Polycarpou, "Effective ACO-based memetic algorithms for symmetric and asymmetric dynamic changes," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 2567–2574, doi: 10.1109/CEC.2019.8790025.

[32] Y. Wu, W. Ma, Q. Miao, and S. Wang, "Multimodal continuous ant colony optimization for multisensor remote sensing image registration with local search," *Swarm Evol. Comput.*, vol. 47, pp. 89–95, Jun. 2019, doi: 10.1016/j.swevo.2017.07.004.

[33] L. Dawson and I. A. Stewart, "Candidate set parallelization strategies for ant colony optimization on the GPU," in *Algorithms and Architectures for Parallel Processing*, vol. 8285. Cham, Switzerland: Springer, 2013, pp. 216–225, doi: 10.1007/978-3-319-03859-9_18.

[34] M. Randall and J. Montgomery, "Candidate set strategies for ant colony optimisation," in *Ant Algorithms*. Berlin, Germany: Springer, 2002, pp. 243–249, doi: 10.1007/3-540-45724-0_22.

[35] M. Sedighpour, V. Ahmadi, M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "Solving the open vehicle routing problem by a hybrid ant colony optimization," *Kuwait J. Sci.*, vol. 41, no. 3, pp. 139–162, Sep. 2014.

[36] J. Peake, M. Amos, P. Yiapanis, and H. Lloyd, "Vectorized candidate set selection for parallel ant colony optimization," in *Proc. Genet. Evol. Comput. Conf. Companion*, Kyoto, Japan, Jul. 2018, pp. 1300–1306, doi: 10.1145/3205651.3208274.

[37] J. Peake, M. Amos, P. Yiapanis, and H. Lloyd, "Scaling techniques for parallel ant colony optimization on large problem instances," in *Proc. Genet. Evol. Comput. Conf.*, Prague, Czechia, Jul. 2019, pp. 47–54, doi: 10.1145/3321707.3321832.

[38] K. Yang, X. You, S. Liu, and H. Pan, "A novel ant colony optimization based on game for traveling salesman problem," *Appl. Intell.*, vol. 50, no. 12, pp. 4529–4542, Dec. 2020, doi: 10.1007/s10489-020-01799-w.

[39] L. Meng, X. You, S. Liu, and S. Li, "Multi-colony ant algorithm using both generative adversarial nets and adaptive stagnation avoidance strategy," *IEEE Access*, vol. 8, pp. 53250–53260, 2020, doi: 10.1109/ACCESS.2020.2967076.

[40] F. Tuani, E. Keedwell, and M. Collett, "Heterogenous adaptive ant colony optimization with 3-opt local search for the travelling salesman problem," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106720, doi: 10.1016/j.asoc.2020.106720.

[41] W. Li, L. Xia, Y. Huang, and S. Mahmoodi, "An ant colony optimization algorithm with adaptive greedy strategy to optimize path problems," *J. Ambient Intell. Human Comput.*, vol. 13, no. 3, pp. 1557–1571, Mar. 2022, doi: 10.1007/s12652-021-03120-0.

[42] P. Wang, Y. Zhang, and D. Yan, "An improved self-adaptive ant colony algorithm based on genetic strategy for the traveling salesman problem," in *Proc. AIP Conf.*, vol. 1967. Busan, South Korea, 2018, Art. no. 40046, doi: 10.1063/1.5039120.

[43] J. Chen, X.-M. You, S. Liu, and J. Li, "Entropy-based dynamic heterogeneous ant colony optimization," *IEEE Access*, vol. 7, pp. 56317–56328, 2019, doi: 10.1109/ACCESS.2019.2900029.

[44] "Multiple depot VRP instances." NEO Research Group. Accessed: Feb. 10, 2022. [Online]. Available: https://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances

[45] C. Contardo and R. Martinelli, "A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints," *Discrete Optim.*, vol. 12, pp. 129–146, May 2014, doi: 10.1016/j.disopt.2014.03.001.

[46] M. E. H. Sadati, D. Aksen, and N. Aras, "A trilevel *r*-interdiction selective multi-depot vehicle routing problem with depot protection," *Comput. Oper. Res.*, vol. 123, Nov. 2020, Art. no. 104996, doi: 10.1016/j.cor.2020.104996.

Petr Stodola received the Ph.D. degree in military technology, informatics, and robotics from the University of Defence, Brno, Czech Republic, in 2006.

He is currently a Full Professor with the Department of Intelligence Support, University of Defence, where he works as a Senior Researcher and a Lecturer. He has authored or coauthored more than 160 scientific papers. His current research interests include optimization, metaheuristic methods, combat modeling and simulation, and command and control decision support systems.

Jan Nohel received the Ph.D. degree in military management from the University of Defence, Brno, Czech Republic, in 2015.

He is currently an Assistant Professor with the Department of Intelligence Support, University of Defence, where he works as a Researcher and a Lecturer. Before starting his research career, he accumulated military experience from being a Commander and the Staff Officer of Czech Republic army units. His current research interests include the information support of the decision-making process of commanders at the tactical command and control level, tactical and terrain analyses, and cross-country movement modeling. He is focused on the issue of mathematical algorithms design for data processing and fusion and their integration in C4ISR systems.