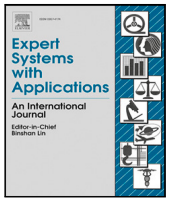
Expert Systems With Applications 251 (2024) 123766

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Management of a post-disaster emergency scenario through unmanned aerial vehicles: Multi-Depot Multi-Trip Vehicle Routing with Total Completion Time Minimization

Tiziana Calamoneri [a,1], Federico Corò [b,1], Simona Mancini [c,d,*,1]

[a] Università di Roma La Sapienza, Roma, Italy
[b] University of Padova, Padova, Italy
[c] University of Palermo, Palermo, Italy
[d] University of Klagenfurt, Klagenfurt, Austria

ARTICLE INFO

Keywords:
Multi-depot
Multi-trip
Completion time
Matheuristic
Unmanned aerial vehicles (UAVs)

ABSTRACT

One of the most valuable and promising applications for Unmanned aerial vehicles (UAVs) is in natural disaster management, where these aircraft can operate autonomously without any need for human intervention during their flights.

In this paper, we foster the interface of Operational Research with computer science in general and sensor networking in particular by focusing on managing a post-disaster emergency scenario where the use of a fleet of UAVs helps rescue teams identify people needing help inside an affected area. We model this situation as an original graph theoretical problem called *Multi-Depot Multi-Trip Vehicle Routing Problem with Total Completion Time minimization (MDMT-VRP-TCT)*. The main novelty of the MDMT-VRP-TCT is the combination of the following three features: multi-depot, multi-trip, and completion time minimization.

We propose a mixed-integer linear programming (MILP) formulation, develop a matheuristic framework to address large instances, and present an extended set of experiments to test the performance of the proposed matheuristic: first, we compare the matheuristic with the MILP formulation on a set of small instances (up to 30 nodes); then, we compare our matheuristic with two heuristics from networking literature, showing that it outperforms the existing algorithms.

1. Introduction

Unmanned aerial vehicles (UAVs) are aircraft that can fly fully autonomously without any human intervention. Initially developed for military purposes, due to advancements in control technologies and decreased costs, UAVs are now being used in a wide range of civilian and commercial sectors, such as forest fire detection (Sharifi et al., 2014), plant disease detection (Gennaro et al., 2016), cargo transport (Thiels et al., 2015), patrolling (Liang et al., 2019), and emergency search and rescue (Calamoneri et al., 2022; Valavanis & Vachtsevanos, 2014). One of the most promising uses of UAVs is in natural disaster management. Numerous papers have been published on this topic recently, *e.g.*, Calamoneri et al. (2022), Cannioto et al. (2017), Erdelj and Natalizio (2016), Erdelj et al. (2017), Estrada and Ndoma (2019), Luo et al. (2019) and Zhan et al. (2018).

This paper focuses on an emergency scenario due to natural disasters such earthquakes, volcanic eruptions, or tsunamis. We propose using a fleet of UAVs to aid rescue teams in identifying people needing help in affected areas. As an example, in Italy, where there are relatively few cities and many small towns and villages, right after an earthquake, typically diverse civil defense rescue teams rush from nearby zones to set up bases around the affected area, each on the road leading to it. For over a decade, civil defense has been able to quickly establish a private broadband emergency wireless network to compensate for the likely disruption of public communication networks. In this way, their bases, scattered around the affected area, can communicate. We can hence assume that the UAVs initially take off from several bases (*multi-depot*), where they return to substitute their batteries and leave

* Corresponding author at: University of Palermo, Palermo, Italy.
E-mail addresses: calamo@di.uniroma1.it (T. Calamoneri), federico.coro@unipd.it (F. Corò), simona.mancini@aau.at, simona.mancini@unipa.it (S. Mancini).
[1] Contributed in equal manner to this work.

https://doi.org/10.1016/j.eswa.2024.123766
Received 18 July 2023; Received in revised form 11 February 2024; Accepted 18 March 2024
Available online 16 April 2024

for a new tour until all affected sites in the disaster area have been flown over. In this context, it is essential to make the most of all the available UAVs, that can opportunely direct rescue teams to possible survivors' locations much faster and more effectively than traditional methods used by civil protection. In particular, UAVs can survey with ease inaccessible sites. Furthermore, they do not require people to control the UAVs (necessary in both the current truck- and drone-based searches typically used by the civil defense), allowing rescue workers to execute other tasks. To ensure that every possible survivor is located, each UAV must travel possibly many times, returning to depots to replace their batteries and continue their search (*multi-trip*) so that the whole area is overflown in the shortest possible time. The ultimate objective is to complete the job in the shortest possible time (*min Total Completion time*, which means the longest completion time among all the UAVs). Hence, we study here the *Multi-Depot Multi-Trip Vehicle Routing Problem with Total Completion Time minimization (MDMT-VRP-TCT)*.

The rest of this paper is organized as follows: after reviewing literature in Section 2, in Section 3 we model MDMT-VRP-TCT as a MILP. Section 4 proposes a matheuristic framework to face reasonably large instances. In Section 5, we experimentally compare its performance first with respect to the exact model (on small instances - up to 30 nodes) and then with respect to two heuristics present in the literature, opportunely modified as the problems they address are not exactly the same (in fact, are special cases of ours, *i.e.* without battery constraints and with a single depot, respectively). Finally, we perform further analyses of the matheuristic. In Section 6, we discuss possible modifications to the problem definition and their consequent impact on the solution; Section 7 concludes the paper by describing some future perspectives.

2. Literature review

Our optimization problem MDMT-VRP-TCT is novel, as its three characteristics can be found in the literature separately but never combined all together. In the following, we survey some problems that have similarities with ours and highlight the essential differences.

2.1. Multi-trip VRP

The *multi-trip* Vehicle Routing Problem (see *e.g.* the survey paper by Cattaruzza et al. (2016)) has been exploited to model specific logistics problems (when electric vehicles or small-sized vans are used — (Kucukoglu et al., 2021) or in the Container Drayage Problem — (Bruglieri et al., 2021), just to cite two applications). The constraints are similar to the ones imposed by the battery endurance of UAVs but beyond the similarities of constraints, the objective function to be minimized is the total cost instead of the completion time.

Regarding the exact approaches, both branch-and-cut (Karaoğlan, 2015) and branch-and-price (Mingozzi et al., 2013) algorithms have been proposed. For what concerns the heuristics, two-stage algorithms (Petch & Salhi, 2003; Taillard et al., 1996) and meta-heuristics, *e.g.*, tabu search (Brandao & Mercer, 1998) and population-based algorithms (Cattaruzza et al., 2014) have been also proposed.

It is worth noting that the battery endurance impose a limitation on the trip length, in Container Drayage, the limitation is directly imposed in the number of nodes per trip. However, in both applications the number of feasible trips is somehow limited, and therefore, the problems can be treated by a trip-based formulation.

With respect to the above mentioned literature, in our paper we minimize the total completion time, while in MTVRP the goal is to minimize the total traveled distance. This makes our problem more complex to address, since while in MTVRP the trips-to-vehicles assignment only influence feasibility, in our case it also impacts the objective function.

2.2. Multi-depot VRP

A problem closely related to MTVRP is the multi-period VRP. While in the former one, the decision maker implicitly decides which customers to serve first, setting the order in which trips assigned to the same vehicle are executed, in the latter, the time horizon is split into several disjointed periods. The decision maker assigns each customer to a period and, based on those assignments, provides a routing plan for each period. A survey paper by Braekers et al. (2016) shows an increasing interest of the researchers especially since 2009 — *e.g.*, (Groër et al., 2009): both exact approaches, *e.g.*, (Dayarian et al., 2015) and meta-heuristics, like Variable Neighborhood Search (Hemmelmayr et al., 2009), and Adaptive Large Neighborhood Search (Mancini, 2016), have been proposed.

The *multi-depot* VRP has been broadly addressed in the literature related to logistics problems in the last three decades. The basic version introduced by Renaud et al. (1996), assigning each vehicle to a depot, has been extended by introducing several additional features, among those: the possibility for vehicles to end a route in a different depot with respect to the one from which it started (Mancini, 2016), the collaboration among different companies, each one owning a subset of the depots (Zhang et al., 2022), the possibility of replenishment at depots during the route (Crevier et al., 2007; Tarantilis et al., 2008). An extended review of multi-depot VRP variants and methods used to address them has been written by Vidal et al. (2011).

Literature on multi-depot multi-trip VRP is scarce. The first paper that simultaneously addresses these two features is by Masmoudi et al. (2016) and studies an application in the dial-a-ride context. Most recently, Zhen et al. (2020) introduce an extension with release dates and time windows, while Sahin and Yaman (2022) consider a heterogeneous fleet of vehicles. All these papers minimize the total traveled distance.

The contribution of this paper to the Multi-depot VRP field is to exploit a completion time minimization objective function, instead of the classical traveled distance minimization.

2.3. Completion time minimization

Almost all works on VRP in the literature consider the minimization of total traveled times or distances, while the minimization of *completion time* (also referred as makespan) has been considered only in relatively few papers. Namely, Archetti et al. (2015) investigate completion time minimization in parcel delivery with release dates, while Poikonen et al. (2017) deal with UAV utilization for last-mile delivery. Talarico et al. (2015) study an application concerning ambulance routing in disaster response. An application in rescue operations is addressed by Calamoneri et al. (2022), who characterize nodes by different levels of priority and aim at minimizing the weighted completion time. The completion time of the longest route is sometimes called *makespan* in the literature. Bakach et al. (2021) study the makespan minimization for the vehicle routing with stochastic travel times, whereas Nadi and Edrisi (2017) provide an application of makespan minimization in relief assessment and emergency response.

In humanitarian applications such as ours, the goal is to reach all the target points as soon as possible, no matter if this goal is reached with a highly unbalanced workload among vehicles. However, in other applications involving human drivers, such as freight delivery, balancing drivers' workload becomes an important issue that cannot be neglected (Mancini et al., 2021).

We are the first to consider the completion time minimization in a multi-depot multi-trip, trip length constrained VRP with a heterogeneous fleet.

2.4. Battery constrained tours

Our problem can be naturally solved in two steps, *i.e.*, first finding tours whose union covers the set of target nodes and secondly determining an opportune scheduling assigning each cycle to a UAV. If we focus on the main subproblem of finding a number of *battery constrained* tours whose union covers the set of target nodes, we find many well-known graph problems in the literature. Namely, RMCCP (Minimum Rooted Cycle Cover Problem) requires finding a bounded rooted cycle cover in which all tours pass through a single depot and have a bounded weight, but it aims at minimizing the number of cycles. Instead, in RMMCCP (Rooted Min–Max Cycle Cover Problem), the optimization function is the completion time, but the maximum number of cycles is bounded by a parameter given in input. These two problems have been proved to be approximable by Frederickson et al. (1976), Friggstad and Swamy (2014), Nagarajan and Ravi (2012) and Yu and Liu (2016). Calamoneri and Tavernelli (2022) establish a connection between the approximability of a problem arising from a UAV application simpler than ours and these two problems.

Our problem strongly differs from both the problems: our goal is to minimize the total completion time, with a limited set of vehicles, while the goal in RMCPP is to cover all the nodes with the minimum number of vehicles possible. Instead, in RMMCCP it is necessary to set the maximum number of cycles, and it is not clear how, if not assigning each vehicle a single trip. Moreover, we allow multiple depots and exploit a heterogeneous fleet, in which different vehicles may have different tour length limitations.

2.5. Applications for UAVs

We observe that the employment of UAVs has grown exponentially in the last few years. There is a diverse set of applications for UAVs, which ranges from civilian (such as logistics, surveillance, and photography) to military use (bomb dropping, war zone medical supply, and enemy spying). The reader is referred to Shakhatreh et al. (2019) for a complete review on UAVs applications. From the extensive literature on this topic in the networking area, we only point out two papers that we will exploit in our experiments: the first one is widely used in the networking area as a benchmark paper, while the objectives of the other one are the most similar to ours. Namely, Kim et al. (2014, 2017) delve into a problem closely aligned with ours. The authors extend the multi-traveling salesperson problem to the domain of multi-UAV path planning in reconnaissance and surveillance scenarios, with a focus on minimizing the longest flight time of the UAVs. They design different variants of the Traveling Salesman Problem with Neighborhood (TSPN) and introduce approximation algorithms to address them. Nevertheless, these works overlook a crucial aspect of real-world applicability, as they fail to account for the limited energy resources inherent to battery-powered vehicles. Calamoneri et al. (2022) introduce the Cover by Multitrips with Priorities problem, introducing a priority value for each site for efficient post-earthquake rescue missions. The authors propose a greedy algorithm that selects cycles by prioritizing sites based on the ratio of priority of the site to distance, ensuring that high-priority locations are inspected promptly while minimizing the overall number of required cycles.

Finally, from a more methodological point of view, algorithms exploiting the combination of trips (or paths) generation procedures and selection/combination mechanism, have been used in different contexts, such as container drayage problems (Bruglieri et al., 2021), last mile distribution in humanitarian relief (Bulcik et al., 2008) and pollution routing (Kramer et al., 2015).

3. Problem definition and mathematical formulation

In this section, we formally describe our application scenario and propose a model in terms of a graph problem.

3.1. Application description

Assume to have an area of interest (*i.e.*, the one affected by a natural disaster) with a set I of n *target nodes* to monitor (*i.e.*, all the damaged buildings). Around this area, there is a set D of *depots* from which a set U of vehicles start and end (*i.e.*, the places where different rescue teams settle down their bases, each with a sub-fleet of UAVs). In general, each vehicle u is equipped with a battery corresponding to b_u units of time (*battery endurance*); when it runs down, it is necessary to replace it with a charged one and, for operational reasons, this can only be done in the depot where every UAV is uniquely associated with o_u. Note that we do not consider the time required for battery recharging and swapping. Indeed, the first task can be performed while the UAVs are in flight while replacing discharged batteries can be done within a few seconds without causing any delays in the takeoff for the next flight. For instance, in Liu et al. (2017, 2018), the authors present a low-cost system that completes the battery swapping process in just 10 s, from landing to takeoff.

3.2. Mathematical formulation

We define a complete edge- and node-weighted graph defined on the node set $N = I \cup D$. Assuming to have the map of the affected area, the *traveling time* between each pair of nodes $i, j \in N$ is known, it is assigned to oriented edge (i, j) as its weight, and is referred to as t_{ij}, expressed in terms of flying time units (assuming, for simplicity, that all UAVs have the same flying speed). Note that the value of t_{ij} could take into account also some variable phenomena, such as the wind; for this reason, in general, we have that $t_{ij} \neq t_{ji}$. A known *service time* s_i is associated as a weight to each node $i \in I$, and represents the needed time to overfly it.

We call *sequence* any ordered set k of target nodes. The *duration* d_k of sequence k is computed as the sum of all traveling times between consecutive target nodes in k plus the service times of all the target nodes in k. In a sequence k, we denote as f_k and l_k the first and the last target nodes, respectively.

The purpose of our problem is to assign to each vehicle $u \in U$ an ordered set of sequences such that u can reach the first target of any of the sequences assigned to it from its depot o_u, serve all its target nodes, come back to o_u, and start again. A sequence k assigned to u with the addition of the depot o_u is called a *trip* and its duration, d_{ku} is given by the traveling distance between o_u and f_k, plus the duration of k, plus the traveling distance between l_k and o_u. A sequence k is *compatible* with a vehicle u if its duration of the associated trip, d_{ku}, is upper bounded by b_u.

A compatibility index, Φ_{ku}, is set to 1 if sequence k is compatible with vehicle u and to 0 otherwise. Of course, k can be assigned to u only if it is compatible with it (*i.e.*, if $\Phi_{ku} = 1$). A sequence k is considered *feasible* if compatible with at least one vehicle. Only feasible sequences are considered. Note that this implies imposing the battery constraint. For each sequence k, we denote by Φ_k the set of all vehicles compatible with k. For each target node $i \in I$, we denote by $\tilde{K}_i$ the set of all feasible sequences containing i. We assume that $\tilde{K}_i$ is not empty for all target nodes, *i.e.*, there is at least one feasible sequence that covers i. If this is not the case, the node is too far to be covered; hence, from now on, we implicitly assume that the depots are located so that the vehicles can reach all target nodes.

A *solution* for our problem consists of selecting a set of sequences K whose union covers I and assigning them to compatible vehicles. The *cumulative working time of a vehicle* u is defined as the sum of the durations of the trips assigned to u. We define the *total completion time of a solution* as the maximum among all the cumulative working times of the vehicles. The goal of our problem is to determine a solution that

minimizes the total completion time. Then, we introduce the following decision variables:

- $X_k \in \{0,1\}$, $k \in K$: binary variable assuming value 1 if sequence k is selected and 0 otherwise;
- $Y_{ku} \in \{0,1\}$, $k \in K$, $u \in U$: binary variable assuming value 1 if sequence k is executed by vehicle u;
- T_u: completion time of vehicle u;
- τ: non-negative variable representing the total completion time.

The mixed integer programming formulation for MDMT-VRP-TCT is the following:

$$\min \quad \tau \tag{of}$$

$$\sum_{k \in \tilde{K}_i} X_k = 1 \quad \forall i \in I \tag{C1}$$

$$\sum_{u \in U} Y_{ku} = X_k \quad \forall k \in K \tag{C2}$$

$$Y_{ku} \leq \Phi_{ku} \quad \forall k \in K \quad \forall u \in U \tag{C3}$$

$$T_u = \sum_{k \in K} d_{ku} Y_{ku} \quad \forall u \in U \tag{C4}$$

$$\tau \geq T_u \quad \forall u \in U \tag{C5}$$

The objective function minimizes the total completion time, as reported in (of). Constraints (C1) ensure that each target node is covered by precisely one sequence. If a sequence is selected, it must be assigned to exactly one vehicle, chosen among those compatible with it (constraints (C2) and (C3)). The cumulative working time for each vehicle is computed by means of constraints (C4). The total completion time must be not smaller than the cumulative working time of each vehicle, as stated in constraints (C5).

The model involves $|K|+|K||U|$ binary variables and $|U|+1$ continuous variables. The number of constraints is $|I|+|K|+|K||U|+2|U|$.

The novelty of the approach used in this formulation consists of generating (open) sequences of nodes that can be assigned to different vehicles at different costs instead of generating trips (that is, sequences plus the depot). It follows that the problem can be modeled as a multiple-choice knapsack, with knapsack-dependent items weight and maximum knapsack occupancy minimization.

It is worth noting that such an approach is not only valid for this specific problem but can be used for a broad class of multi-depot multi-trip problems, including those having different objective functions, such as the classical total traveling distance minimization, the minimization of the number of vehicles used, or the minimization of the total cost given by vehicles purchasing costs plus travel costs, as commonly in use in logistics applications.

4. A model-based matheuristic framework

The main idea under the mathematical model consists of generating all possible feasible sequences, and associating them with the set of their compatible UAVs. When the number of feasible sequences is too large to be handled, the mathematical model becomes intractable. If, for instance, target nodes are so close to each other that a huge number of feasible sequences are produced, or batteries are so large that several target nodes can be visited in a single sequence, even on instances with less than 10 nodes.

To overcome this issue and address larger instances, we derive from our model a heuristic approach, in which we generate only a subset of feasible sequences $\tilde{K}$ to be passed to the model. Clearly, the choice of sequences can dramatically change the performance of the heuristic. Therefore, the problem of determining which sequences to generate is of crucial importance.

4.1. Generation of promising sequences

In the following, after giving operative definitions, we describe how we generate promising sequences to be passed to the mathematical model.

Given a sequence, we call its *extremes* the first and the last node in any order. For example, sequence {1,2,3,4,5} has the same extremes as sequence {5,3,4,2,1}. A sequence k is *dominated* by another k' if they have the same extremes and contain precisely the same target nodes (possibly in a different order), but k' has a lower or equal duration than k. A sequence k is *strictly dominated* by another k' if k is dominated by k' and they do not have the same duration.

The heuristic, whose pseudo-code is detailed in Algorithm 1, incrementally constructs longer and longer sequences, starting from shorter ones. To do this, it receives in input two parameters N_c and K_{max}; N_c represents the number of sequences generated starting from a shorter and already generated one, while K_{max} is the maximum allowed number of sequences. It initially generates all the sequences that contain only one target node (Line 5) and inserts them directly into the set $\tilde{K}$ of sequences to be passed to the model (Line 6). It also inserts them in a temporary queue K^{tmp}, containing sequences to be expanded (Line 2). Namely, every sequence k included in K^{tmp} is processed as follows:

- a sequence k is dequeued from K^{tmp} (Line 10) and N_c child sequences are generated each one by extending sequence k with an additional target node. These N_c nodes are selected from the nearest nodes to the last target node of sequence k, among those that have not yet been included in k (Lines 12–13).
- For each child sequence k^c, two feasibility checks are applied: if k^c is compatible with at least one vehicle, and if the sequence neither is strictly dominated by nor strictly dominates another sequence already belonging to $\tilde{K}$ (Line 14). Observe that each new child sequence can dominate at most one of the solutions in $\tilde{K}$. (In fact, if two solutions exist, k' and k'', sharing the same target nodes and extremes with k and, without loss of generality, $d_{k'} \leq d_{k''}$, k' would be strictly dominated by k'' and therefore it could not belong to $\tilde{K}$.)
- If a feasibility check fails, the sequence is discarded, otherwise it is added to $\tilde{K}$ and enqueued to K^{tmp} (Lines 15–16).

The procedure terminates either when K^{tmp} is empty or when a number K_{max} of sequences have been added to $\tilde{K}$ (Line 9). After the sequence generation process is finished, the set of sequences $\tilde{K}$ is given in input to the mathematical model (Eq. (of)–(C5)) to find the best solution obtainable with the subset of trips provided (Line 24). It is worth noting that each target node, also the most isolated one, appears in some sequences of $\tilde{K}$ (at least it is in as a sequence constituted by a single target node, and it appears in all the sequences generated starting from it).

We note that the parameter K_{max} plays a crucial role in the algorithm performance: lower values will give worse solutions while higher values will yield a better global solution but increase the computational time required by the heuristic. This parameter must be properly tuned to achieve the right balance between solution quality and computational time. In Section 5 we deeply discuss the tuning of K_{max}.

The maximum number of children generated by each sequence, N_c, also plays an important role. The higher the value of N_c, the higher the number of sequences containing a specific number of target nodes. Note that by fixing the value of K_{max}, lower values of N_c allow us to generate sequences containing more target nodes, which could be promising; on the other hand, in those sequences, nodes that are closely located to each other would tend to be visited more frequently, while isolated targets would appear in only few sequences. Instead, with large values of N_c, even targets that are more widely separated can be visited. Still, the maximum allowed number of sequences K_{max} would be reached even only with sequences that contain a small number of targets. Longer sequences would not be generated, with a negative

effect on the solution quality. For example, with $N_c = 10$, we would have 10 trips containing only one target. Each of these trips generates 10 trips with 2 targets, and so on. Hence, the number of generated trips with low cardinality (low number of targets visited) is high, and as a consequence, the maximum threshold K_{max} is already reached before starting to generate high cardinality trips. On the contrary, when N_c is small *e.g.*, when $N_c \leq 3$, the number of trips generated by each trip does not grows up fast, and therefore, we can generate longer sequences before reaching K_{max}. In conclusion, it is essential to carefully tune the values of parameters K_{max} and N_c together to achieve a balance between diversity among sequences and sequence length.

Algorithm 1 A model-based matheuristic

Input: I, N_c, K_{max}
Output: A feasible solution, *i.e.*, a set of trips overall covering all target nodes, each assigned to a vehicle

set of sequences $\tilde{K} = \emptyset$
queue $K^{tmp} = \emptyset$
for all $i \in I$ **do**
    Restricted Candidate List of i (RCL_i) := list of N_c closest nodes to i
    k_i := sequence constituted by sole node i
    $\tilde{K} = \tilde{K} \cup \{k_i\}$
    enqueue(K^{tmp}, k_i)
end for
while $K^{tmp} \neq \emptyset$ AND $|\tilde{K}| < K_{max}$ **do**
    sequence k = dequeue(K^{tmp})
    l_k := last node in sequence k
    for all $c \in RCL_{l_k}$ **do**
        $k^c := k \cup c$
        if k^c is feasible and not dominated by any sequence in $\tilde{K}$ **then**
            $\tilde{K} = \tilde{K} \cup \{k^c\}$
            K^{tmp} = enqueue(K^{tmp}, k^c)
            remove from $\tilde{K}$ all the sequences strictly dominated by k^c
            for all $i \in k^c$ **do**
                update RCL_i
            end for
        end if
    end for
end while
feed the mathematical model defined by equations (C1)-(C5) and (of) with all the paths in $\tilde{K}$ and solve it.

It is worth noting that sequences of different sizes are particularly suitable for multi-trip problems when dealing with completion time minimization, because having items of heterogeneous size helps to equalize the jobs of the vehicles. Indeed, while in a classical multi-trip problem, the trips-to-vehicles assignment phase only impacts feasibility, when we aim to minimize total completion time, it also influences the objective function. For this reason, in our case, it is essential to have trips of different sizes to better balance the global workload.

If we compare the trip generation procedure used by the Greedy randomized adaptive search (GRASP), (Resende & Ribeiro, 2010), with ours, we realize that the first one tends to generate trips almost of the same length, aiming to fully utilize the vehicle capacity, while the second produces trips of different lengths, from very short to very long. If the approach exploited by GRASP is certainly profitable when the goal is to minimize the total travel distance/time (Layeb et al., 2013), ours offers more combination options when performing the assignment to vehicles with the objective of minimizing completion time.

Moreover, our method is more suitable even for problems with a heterogeneous fleet because short trips can be exploited for vehicles with more limited endurance, while the GRASP generation approach better works with a homogeneous fleet (Layeb et al., 2013), as tends to generate sequences that exploit the whole endurance/autonomy of the vehicle.

5. Computational results

In this section, we study the performance of our matheuristic. Namely, first, we compare the matheuristic with the exact model on a set of small instances (up to 30 nodes) to test the behavior of our optimization approaches.

In particular, the metaheuristic finds results up to 4% better than the MILP — that needs to be interrupted to guarantee a reasonable running time and hence is not able to provide an optimum solution; on top of that, it improves in computational efficiency by three orders of magnitude.

Secondly, we compare our matheuristic with two state-of-the-art heuristics from networking literature. Also, in this case, we show that it outperforms the existing algorithms, paying a bit in terms of computational time that is kept at absolutely reasonable levels.

All our experiments have been performed on a computer equipped with an Intel(R) Core(TM) i5-1135G7 CPU (8 cores clocked at 2.4 GHz) and 16 GB RAM; our programs have been implemented in C++ (g++ compiler v9.4.0 with optimization level O3).

The area of interest is set as a square with a side length equal to 15 km×15 km, and the depots are positioned on a subset of its 4 vertices. The target nodes are randomly positioned inside it, and their number n moves from $n = 10$ to $n = 200$.

While considering different values for the number of vehicles, target nodes, and depots, we run our experiments on combinations of the following values: $U = 2, 9, 12, 15, 20$ and $D = 2, 3, 4$. In the following subsections, we show the results for each setting.

The K_{max} value is set to 50000 in all the experiments, allowing the metaheuristic to solve the instances in efficient computational times.

5.1. Comparison with the model

Here, we compare our matheuristic with the exact model when there are two depots and one vehicle per depot, with 30 and 50 min of battery endurance, respectively. We perform two sets of experiments, one with service times of the target nodes randomly chosen in the interval $(5, 8]$ (Fig. 1) and another one with service times randomly chosen in the interval $(0, 3]$ (Fig. 2); the reason is that – as we have already pointed out in Section 4 discussing the tuning of K_{max} and N_c – when the service times are long, each trip contains fewer target nodes than when the service times are short, so obtaining different performance for our matheuristic as n grows up.

In all charts, on the x axis, 3, 4, 5, and 6 represent the used values of N_c. The y coordinates of the dots correspond to an average computed on 20 random instances on the same number of nodes: every column of charts corresponds to a different value of n (increasing going from left to right). The red lines represent the benchmark values achieved by the model. It is worth noting that when n is small ($n \leq 30$ in Fig. 1 and $n \leq 10$ in Fig. 2), the model can produce exact results; when n is larger, the model terminates only in a few cases (probably when the instances are particularly easy to solve, *e.g.*, if they have no clustered target nodes). Note that when service times are longer (Fig. 1), the model can handle instances with higher values of n because the produced sequences contain fewer nodes than in the case with shorter service times, and therefore their number is more tractable.

The experiments perfectly confirm the expectations. Indeed:

First row. The first three charts of Fig. 1 and the first one in Fig. 2 show the percentage gap between the heuristically computed completion time and the optimum value, which is the primary objective function of our problem; it is clear that the percentage gap decreases as N_c grows up and, when $N_c = 6$, it gets close to 0, indicating the effectiveness of

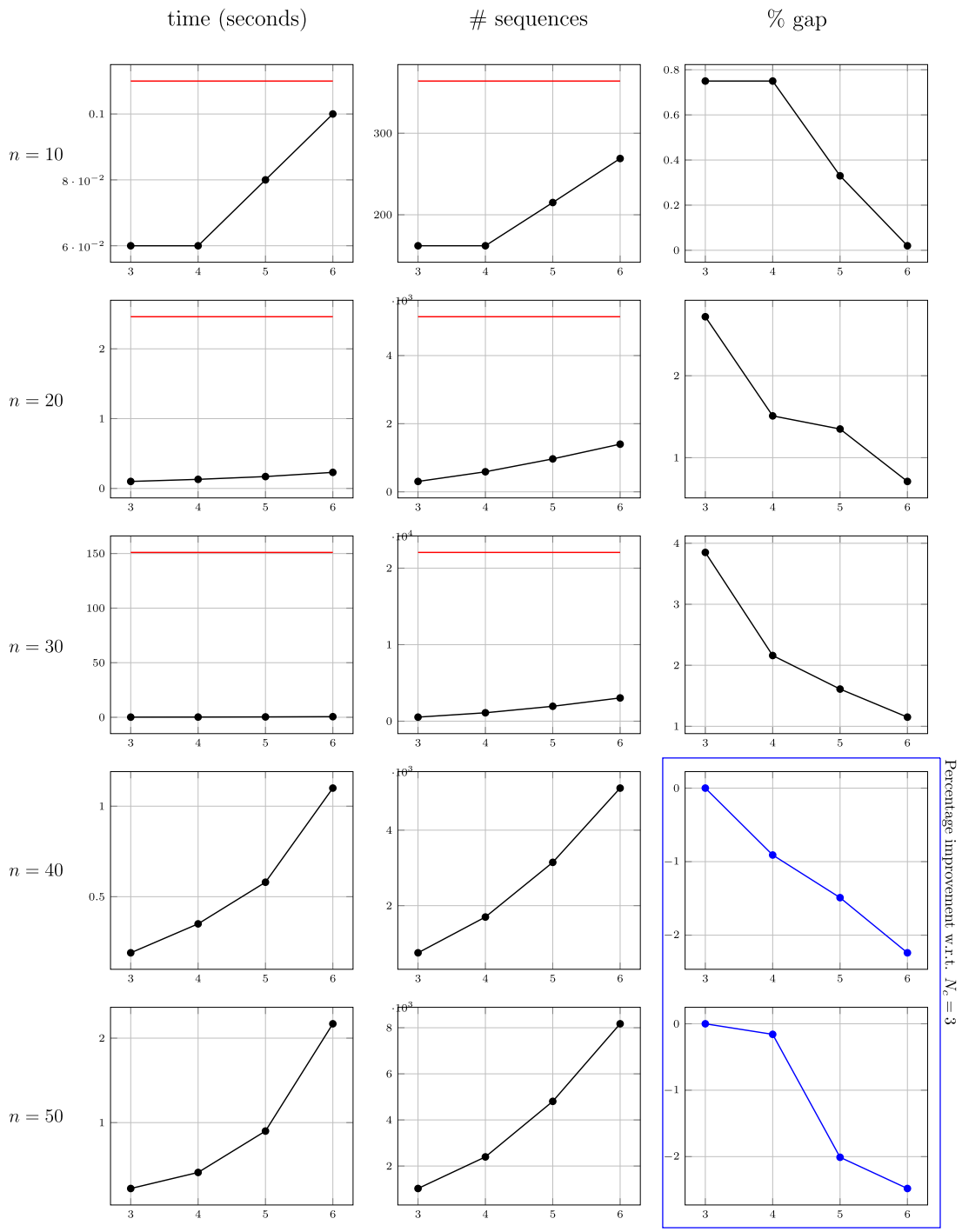


Fig. 1. Experimental results with service times randomly chosen in the interval $(5, 8]$. On the x axis, 3,4,5,6 represent the used values of N_c; K_{max} is set to 50000; the red lines represent the benchmark values achieved by the model.

our matheuristic. Since there is no benchmark given by the exact model when $n \geq 40$ in Fig. 1 and when $n \geq 20$ in Fig. 2, the rightmost charts of the first row (in blue) show the percentage gaps with respect to the case $N_c = 3$; these gaps are negative since clearly large values of N_c lead to better solutions. This is not always true in Fig. 2; the reason is that every trip includes many target nodes, therefore a large number of feasible sequences are generated, and the value of K_{max} is reached, and the quality of the solution degrades as N_c increases.

Second row. Here, we depict the number of trips generated to identify the solution. As expected, the matheuristic generates a limited number of trips, making the approach viable even for large instances. Note that the number of trips is higher in Fig. 2 than in Fig. 1 because shorter service times imply trips with a larger number of nodes, and consequently overall fewer trips. In Fig. 2, it is evident when the value of K_{max} is reached, as the plot of the function becomes horizontal.

Third row. The running times are reported here. Clearly, the computational time of the model is much higher and, for what concerns the matheuristic, it grows up as the value of N_c increases in Fig. 1. However, it remains at least one order of magnitude smaller than the time necessary for the model. Also, the computational time is much

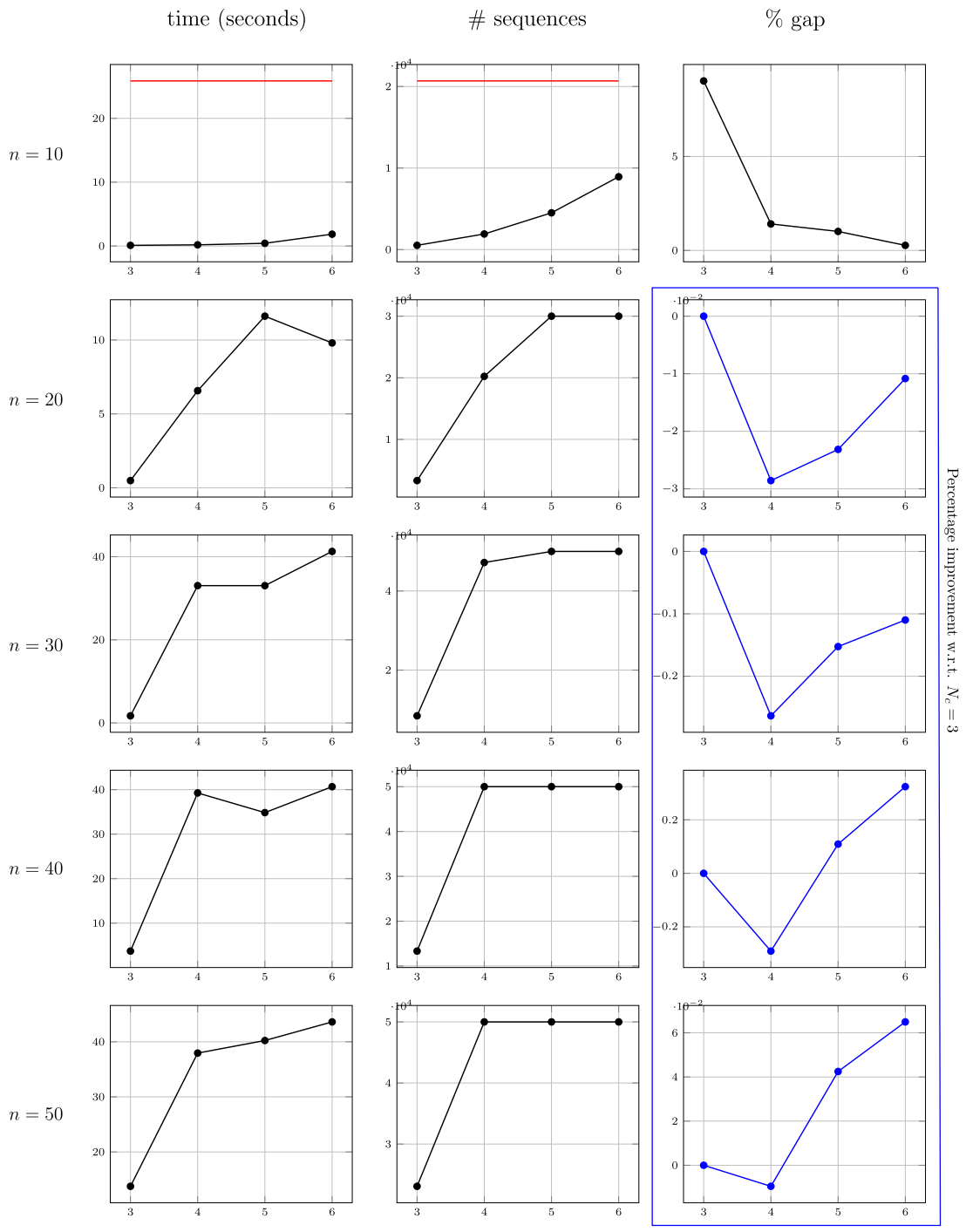


Fig. 2. Experimental results with service times randomly chosen in the interval $(0, 3]$. On the x axis, 3,4,5,6 represent the used values of N_c; K_{max} is set to 50000; the red lines represent the benchmark values achieved by the model.

higher in Fig. 2 than in Fig. 1. Still, it does not continue to grow with N_c because again when K_{max} is reached, the computational time remains about constant.

We chose the value of N_c based on the following observation. The first line of Fig. 2 shows that, when $n > 10$, the best results are obtained for $N_c = 4$, *i.e.*, the gap percentage is minimum, but are not so far from the case $N_c = 3$ (while $N_c = 5$ and $N_c = 6$ are clearly outperformed). On the contrary, increasing the size of the instance, the gap between $N_c = 3$ and $N_c = 4$ becomes smaller and smaller. The high gap shown between the two cases $N_c = 3$ and $N_c = 4$ on small instances is because, when the number of nodes in the network is small ($n = 10$), the number of trips generated with $N_c = 3$ is too small (a few hundred); however, in such instances, the exact model works in a short computational time, hence the usage of heuristics is not needed in these cases.

On the other hand, if we look at the third row (computational times), the case $N_c = 3$ is about one order of magnitude faster than $N_c = 4$, and therefore it is preferable. This suggests that, in real-size instances, the case $N_c = 3$ represents the best compromise between heuristic performance and computational times. Hence, it is our choice for further comparison with other heuristics.

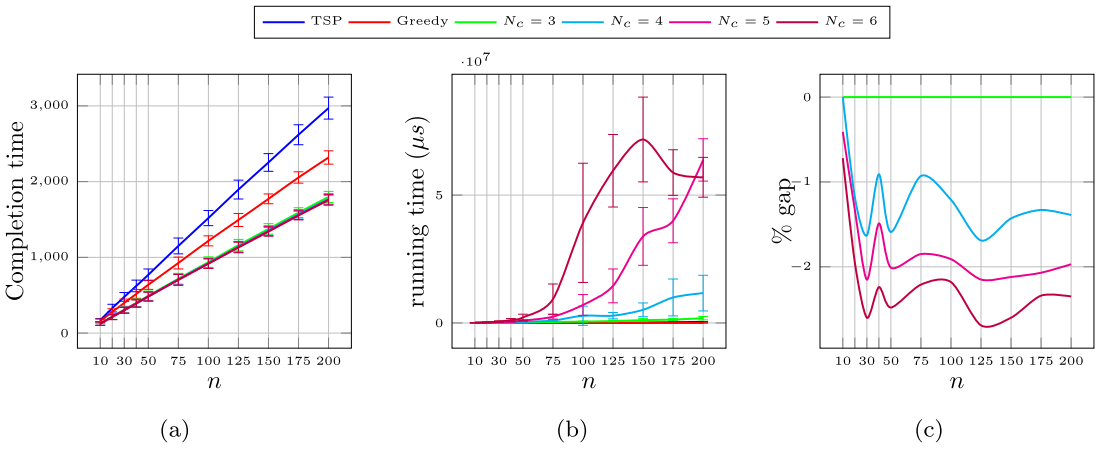


Fig. 3. (a) completion times; (b) running times; (c) percentage improvement with respect to $N_c = 3$. In these experiments, $|D| = 2$, $|U| = 2$ and service times are randomly chosen in the interval $(5, 8]$.

5.2. Comparison with other heuristics

We now compare the performance of our matheuristic with two benchmark state-of-the-art heuristics known in the literature in the networking field. They do not solve specifically our problem, which is new, but consider a setting that is sufficiently close to ours to allow us to fairly modify them to let them work in our scenario.

More in detail, on the one hand, we consider one of the most widely used algorithms in the networking area designed in Kim et al. (2014, 2017) (called from now on $\mathcal{H}_{TSP}$), solving a problem similar to ours except that there are no battery constraints. On the other hand, we exploit one of the heuristics from Calamoneri et al. (2022) (called from now on $\mathcal{H}_{Greedy}$) facing a scenario that is very similar to ours, except that a single-depot model is followed and priorities are introduced.

The original $\mathcal{H}_{TSP}$ algorithm gives as part of the input the vehicles, each one positioned on its own depot, and uses them as children of the root of a minimum spanning tree rooted at a dummy node v_0; the subtrees rooted at the vehicles are then transformed into trips covering all target nodes and intersecting only in the root using the Christofides's approximation algorithm for TSP (Christofides, 1976); finally, some operations are executed to equalize the weight of the different trips. We modify this algorithm by imposing that the duration of each cycle is kept upper bounded by the battery endurance of the UAV to which it is assigned. Then, all the served target nodes are removed from the graph, and the algorithm is iteratively re-run on the remaining graph until it is empty.

Heuristic $\mathcal{H}_{Greedy}$ exploits a greedy approach to compute in parallel as many cycles as the number of vehicles. The problem solved by $\mathcal{H}_{Greedy}$ considers priorities assigned to the target nodes, and the heuristic considers them when making the greedy choice. Here, we do not have any priority and simplify the computation: for each UAV u, at each step, the current partial trip associated with it is considered and, starting from the target node selected last (at the beginning, the depot), the next target is chosen as the closest one if it still guarantees that the whole trip with the addition of this last target node can be flown over by UAV u within the battery endurance b_u. The fact that the original addressed problem had a single depot does not change the heuristic. We highlight that our modifications to the original algorithms do not compromise their performance in any aspect but are only meant to extend their applicability to our scenario.

In all following charts, x coordinates represent increasing values of n, from 10 to 200; moreover, we label lines with 'TSP' and 'Greedy' to mean that they are referred to $\mathcal{H}_{TSP}$ and $\mathcal{H}_{Greedy}$, respectively. Regarding our matheuristic, we label the lines with '$N_c = i$', where $i = 3, 4, 5, 6$ to distinguish the value used for this parameter.

Fig. 3(a) shows the completion times computed on 20 instances on the same number of nodes obtained with $\mathcal{H}_{TSP}$, $\mathcal{H}_{Greedy}$ and our matheuristic when $N_c = 3, 4, 5, 6$ in the special case in which there are two depots, each one with one vehicle. The matheuristic achieves the best results. Fig. 3(b) shows the average running times (microseconds), and in this case, the two benchmark heuristics are faster. It is worth noting that the function corresponding to the execution time of our matheuristic when $N_c = 6$ is not increasing; this can be explained because, for sufficiently large values of n, the threshold K_{max} bounding the maximum number of trips has been reached. Hence, the computational times do not grow up anymore. Moreover, of course, for our matheuristic, higher running times correspond to larger values of N_c but, in Fig. 3(a) the completion times appear to be similar; hence, in Fig. 3(c), we depict the percentage of improvement of the completion time when passing from $N_c = 3$ to the larger values of N_c. The improvement in the quality of the solution, when N_c is larger than 3, of at most 2.7% confirms what we observed at the end of Section 5.1, justifying that, in the following charts, we compare the benchmark heuristics $\mathcal{H}_{TSP}$ and $\mathcal{H}_{Greedy}$ with our matheuristic only with $N_c = 3$.

For the following experiments, we assume that the sub-fleet based at each depot is homogeneous, although the whole fleet is non-homogeneous, in agreement with the inspiring application, where each depot is supervised by one rescue team, that we suppose to own a fleet of identical UAVs. The battery endurance is set to 50, 30, 40, and 20 min for all the vehicles based at the first, second, third, and fourth depots, respectively (whenever present), to guarantee that there are always vehicles whose endurance allows them to reach any target in the area of interest.

In Fig. 4 we compare the benchmark heuristics with our matheuristic with $N_c = 3$ varying in different ways the values of $|D|$ and of $|U|$ with service times randomly chosen in $(5, 8]$; we choose not to consider when the service times are in $(0, 3]$ because, as already observed, the number of generated trips soon reaches the value of K_{max}.

More in detail:

- in Fig. 4(a), two depots positioned onto two adjacent vertices of the area of interest, base of 6 and 3 vehicles, respectively;
- in Fig. 4(b), two depots positioned onto two opposite vertices of the area of interest, both base of 10 vehicles;
- in Fig. 4(c), three depots, each base of 3 vehicles;
- in Fig. 4(d), three depots, base of 6, 4, and 2 vehicles, respectively;
- in Fig. 4(e), four depots, each base of 3 vehicles;
- in Fig. 4(f), four depots, base of 8, 4, 2, and 1 vehicle, respectively.

Therefore, results show that our heuristic outperforms the benchmarks for the studied set of instances.

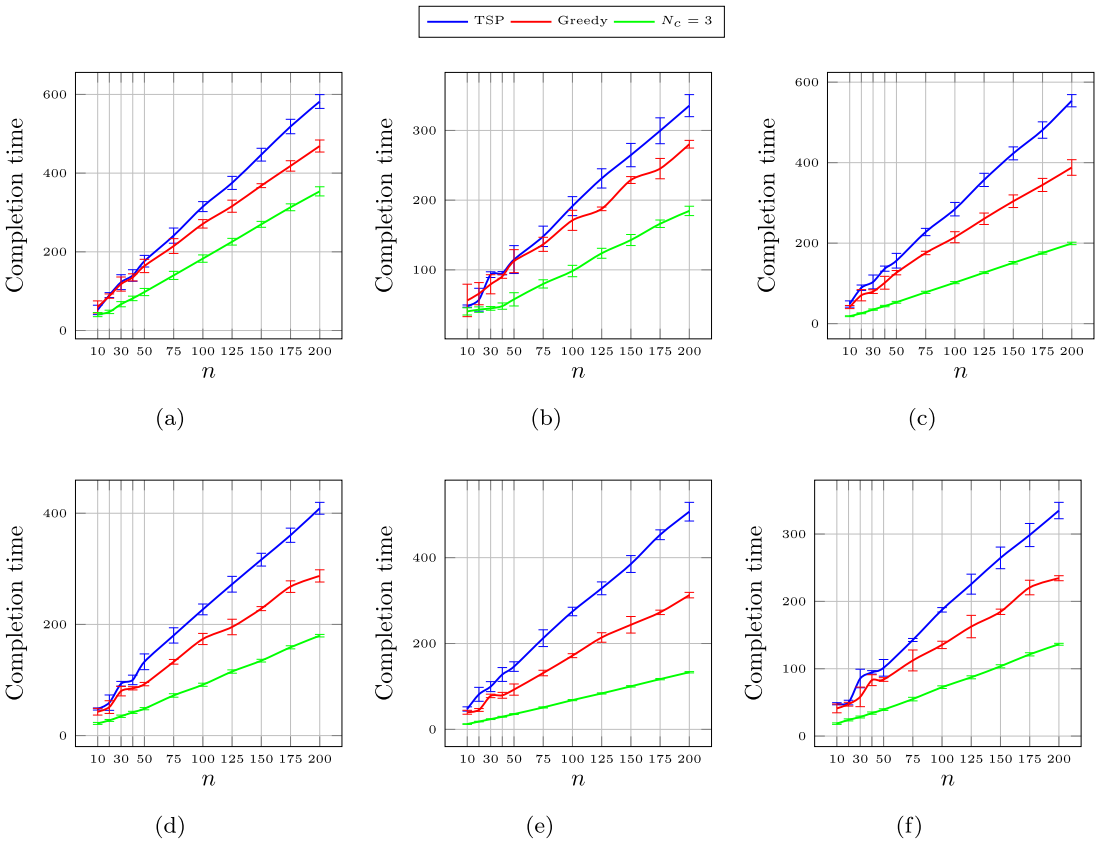


Fig. 4. Completion times when service times are randomly chosen in the interval $(5, 8]$. The number of depots and of UAVs is different in each chart: (see details in the text).

6. Discussions on the definition of the problem

In this section, we discuss possible modifications to the problem definition and their consequent impact on the solution.

6.1. Completion time vs. travel distance minimization

While in logistics the most relevant aspect is cost minimization, in the humanitarian context, completion time plays a crucial role. As already pointed out, the problem studied in this paper can be defined as a multi-trip multi-depot VRP with total completion time minimization (*i.e.*, the minimization of the completion time of the UAV which finishes its tasks last). It differs from classical multi-trip multi-depot problems, where the total travel distance (or time) is minimized. By minimizing the traveling distance, one expects to minimize the completion time as well. This is in general not true. As an example, consider the instance depicted in Fig. 5, where the area of interest is a square with a unit side and the budgets of the two vehicles are assumed to be enough to traverse 2 distance units each; the best solution for optimizing the completion time is shown in Fig. 5(a), while the best solution for optimizing the total traveling distance is different and is shown in Fig. 5(b).

Although this is an artificial example, it hides a more general behavior. Indeed, in the following we experimentally compare the results output by the exact model in the two cases in which the objective function is the completion time (Eq. (of)) and the total traveling distance. We run our model on 20 random instances with 20 target nodes whose service times are between 5 and 8 min, 2 depots, and a single vehicle *per* depot, one with a battery endurance of 30 min and the other UAV with a battery endurance of 50 min; when the objective is minimizing the completion time, we get an average value of 211.46 with an average number of trips equal to 12.15, while the traveling distance computed *a posteriori* is 397.40; vice-versa, when we aim at minimizing the traveling distance we get an average value of 333.70 with an average number of trips of 8.25, while the completion time computed as a consequence is 283.40. We deduce that, although the two parameters are clearly not uncorrelated, there is neither a clear dependence. More specifically, minimizing the completion time, the traveling distance is slightly higher than its optimum value (about 19%), but when we minimize the traveling distance, the completion time grows up much more (about 34%).

Note that the minimization of the completion time is associated with a more significant number of trips, while minimizing the total distance requires to have fewer trips in the solution. The reason is that when the completion time is minimized, it is better to equalize the flying time of each vehicle, even at the cost of producing a larger number of trips. Vice-versa, if the traversed distance is minimized, the best choice is to produce few and maximally full trips, even at the cost of making a vehicle work much more (this is more evident when, as in this case, the battery endurance of the UAVs are rather different).

6.2. Multi-depot vs. single-depot

The definition of our problem as a multi-depot one comes from the real-life problem we started from; indeed, several rescue teams may arrive from different directions and fix their bases in different locations. Nevertheless, it is interesting to analyze the benefit achievable by exploiting multiple depots with respect to a single depot. We compare a solution when a single depot is located in the bottom-left corner of a squared area, with a solution of the same instance when 4 depots are located in each corner. We perform three analyses:

(1) target points are uniformly distributed across the whole area;

(2) target points are uniformly distributed in the up-right quadrant (*i.e.*, far from the single depot);

(3) target points are uniformly distributed in the bottom left quadrant (*i.e.*, near the single depot).

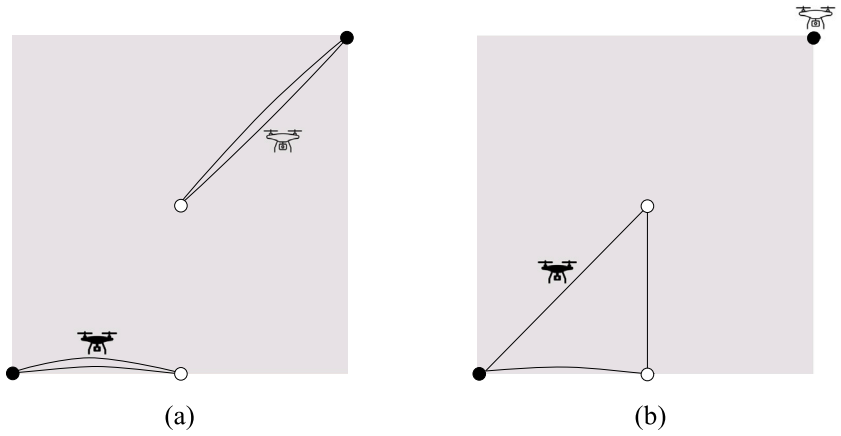


Fig. 5. (a) a solution minimizing the completion time; (b) a solution minimizing the traveling distance. Black dots represent the depots while white dots represent targets.

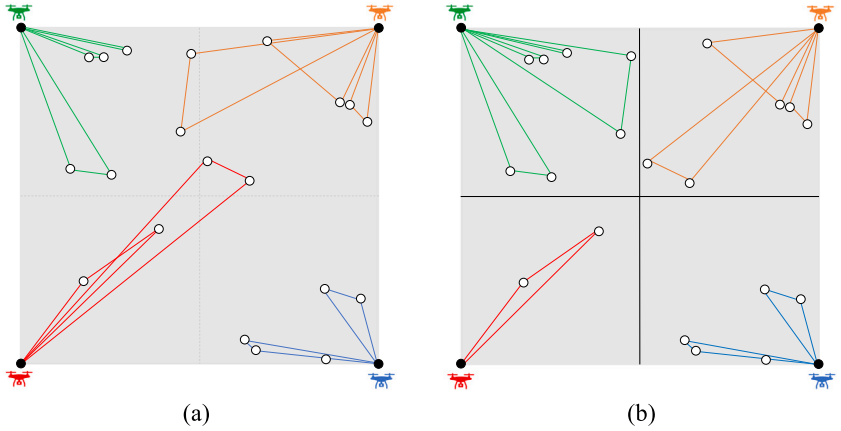


Fig. 6. (a) An optimal solution on a sample instance (Completion Time: 41.28); (b) a solution where a different quadrant is exclusively assigned to each UAV. (Completion Time: 65.41).

As expected, the single-depot approach performs better than the multi-depot one, when target points are concentrated near it (case 3). However, the gain is quite limited (13%). Instead, the gains achieved with a multi-depot approach in cases 2 and 1 are around 53% and 39%, respectively, making this approach globally preferable.

6.3. Considering the area as a whole vs. as partitioned among depots

Finally, we could think that our problem would be easier if we partition the area of interest into as many portions as the number of depots so that the target nodes falling in a certain portion are automatically assigned to the closest depot. We perform experiments where we consider a symmetric situation for what concerns depots and vehicles (*i.e.*, 4 depots, one vehicle *per* depot, all with the same battery endurance) in order to partition the squared area of interest into 4 equal sub-squares. In agreement with the intuition, the computational time is shorter (0.1 secs versus 1 s) but the difference between those computational times is not practically relevant. On the other hand, the completion time is much worse (on average of about 45%); this can be explained because an optimal solution could require that a vehicle enters the sub-square assigned to a different depot in order to drain its battery. Moreover, since the goal of the problem is to minimize the total completion time, it is advantageous to distribute the workload to the different UAVs almost homogeneously. Permitting vehicles to serve nodes located outside their own area allows them to balance the workload better and reduce completion time.

Fig. 6 shows the behavior of a sample instance: in Fig. 6(a) (where all UAVs can fly over every target) the trips are assigned in a balanced fashion; on the contrary, in Fig. 6(b) (where a subarea is exclusively assigned to each vehicle), the UAV with the base at the lower-left vertex is scarcely used (only one trip is assigned to it); vice-versa, 4 trips are assigned to the UAV with the base at the upper-left vertex, lengthening the completion time; note that the UAV with base on the lower-right vertex overflies precisely the same target nodes in both the scenarios, as they are sufficiently close to it and are anyway the best choice.

7. Conclusions and future perspectives

In this paper, we considered a real-life situation modeled as a multi-depot multi-trip routing problem where we aimed at minimizing the total completion time.

We have pointed out that the problem we introduce has similarities to other problems in the literature but essentially stands out for each of them, giving rise to a new mathematical formulation.

We proposed for our problem a formulation as a MILP, designed a matheuristic framework to solve large instances quickly and presented an extended experimental campaign that shows the potential of the proposed matheuristic. First, we compared the MILP formulation with the metaheuristic on a set of small instances (up to 30 nodes) to test the optimization approaches performance.

Then, we compared our matheuristic with two heuristic approaches derived from state-of-the-art methods for similar existing problems from networking literature and checked that the matheurstic we proposed outperforms both of them. From these experiments, we deduced the best values of the two input parameters, K_{max} and N_c, to be set to 50000 and 3, respectively.

Beyond the proposed application, our solution approach, based on sequences that become trips only when they are assigned to vehicles, can be extended to any case in which a multi-depot multi-trip vehicle routing problem must be solved.

From the discussions on the problem definition, we derive the following managerial insights. Namely, when the set of target nodes is randomly distributed across an area, it is more profitable to exploit multiple depots, instead of one, from which to launch UAVs. Moreover, it is better that all vehicles cooperate to overfly all target nodes, instead of partitioning the area among the depots.

Concerning future work, many interesting generalizations can be introduced to make our model more flexible for practice. Namely, first, we handled the multi-depot model, assigning each UAV once and for all to a depot. This matches our real-life application because it is reasonable that every rescue team brings its own fleet of UAVs to the site it chooses as its depot and is the most suitable to manage it. We could relax this condition so that, looking at the ordered sequences assigned to each UAV, we only require that a sequence's starting depot coincides with the previous sequence's arrival depot. This would introduce more flexibility, but as UAVs recharge/change their batteries at the depot, it would be required to ensure that there is either a free recharge station or a charged battery available in the chosen depot before landing. Secondly, we could make the model more general, adding further variables that would make the model closer to the real-life application. Namely, as already proposed by Calamoneri et al. (2022), we could introduce uncertain service times and priorities on the target nodes. The first one is useful when, thanks to a computing phase on board, a UAV realizes that there are possible survivors to save in correspondence with a certain target node and decides to spend more time flying over the target. The second one makes sense if we want to fly over certain buildings (*e.g.*, schools and hospitals) before others.

Declaration of competing interest

None

Data availability

Data will be made available on request.

References

Archetti, C., Jabali, O., & Speranza, M. G. (2015). Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, *61*, 122–134.

Bakach, I., Campbell, A., Ehmke, J., & Urban, T. (2021). Solving vehicle routing problems with stochastic and correlated travel times and makespan objectives. *EURO Journal on Transportation and Logistics*, *10*, Article 100029.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300–313.

Brandao, J., & Mercer, A. (1998). The multi-trip vehicle routing problem. *Journal of the Operational research society*, *49*, 799–805.

Bruglieri, M., Mancini, S., Peruzzini, R., & Pisacane, O. (2021). The multi-period multi-trip container drayage problem with release and due dates. *Computers & Operations Research*, *125*.

Bulcik, B., Beamon, B., & Smilowitz, K. (2008). Last mile distribution in humanitarian relief. *Journal of Intelligence Transportation Systems*, *12*(2), 51–63.

Calamoneri, T., Corò, F., & Mancini, S. (2022). A realistic model to support rescue operations after an earthquake via uavs. *IEEE Access*, *10*, 6109–6125.

Calamoneri, T., & Tavernelli, D. (2022). Modeling and approximating the visit of a set of sites with a fleet of UAVs. *The Computer Journal*.

Cannioto, M., D'Alessandro, A., Lo Bosco, G., Scudero, S., & Vitale, G. (2017). Brief communication: Vehicle routing problem and uav application in the post-earthquake scenario. *Natural Hazards and Earth System Sciences*, *17*, 1939–1946.

Cattaruzza, D., Absi, N., & Feillet, D. (2016). Vehicle routing problems with multiple trips. *4OR. A Quarterly Journal of Operations Research*, *14*, 223–259.

Cattaruzza, D., Absi, N., Feillet, D., & Vidal, T. (2014). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, *236*, 833–848.

Christofides, N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem*: *Technical Report*, Carnegie-Mellon Univ.

Crevier, B., Cordeau, J.-F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, *176*, 756–773.

Dayarian, I., Crainic, T. G., Gendreau, M., & Rei, W. (2015). A branch-and-price approach for a multi-period vehicle routing problem. *Computers & Operations Research*, *55*, 167–184.

Erdelj, M., & Natalizio, E. (2016). UAV-assisted disaster management: Applications and open issues. In *2016 international conference on computing, networking and communications* (pp. 1–5).

Erdelj, M., Natalizio, E., Chowdhury, K. R., & Akyildiz, I. F. (2017). Help from the sky: Leveraging UAVs for disaster management. *IEEE Pervasive Computing*, *16*, 24–32.

Estrada, M. A. R., & Ndoma, A. (2019). The uses of unmanned aerial vehicles UAVs-(or drones) in social logistic: Natural disasters response and humanitarian relief aid. *Procedia Computer Science*, *149*, 375–383, ICTE in Transportation and Logistics 2018 (ICTE 2018).

Frederickson, G. N., Hecht, M. S., & Kim, C. E. (1976). Approximation algorithms for some routing problems. In *17th annual symposium on foundations of computer science* (pp. 216–227).

Friggstad, Z., & Swamy, C. (2014). Approximation algorithms for regret-bounded vehicle routing and applications to distance-constrained vehicle routing. In *Proceedings of the 46th annual symposium on theory of computing* (pp. 744–753).

Gennaro, S. D., Battiston, E., Marco, S. D., Facini, O., Matese, A., Nocentini, M., Palliotti, A., & Mugnai, L. (2016). Unmanned aerial vehicle UAV-based remote sensing to monitor grapevine leaf stripe disease within a vineyard affected by esca complex. *Phytopathologia Mediterranea*, *55*.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & service operations management*, *11*, 630–643.

Hemmelmayr, V., Doerner, K., & Hartl, R. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, *195*, 791–802.

Karaoğlan, I. (2015). A branch-and-cut algorithm for the vehicle routing problem with multiple use of vehicles. *International Journal of Lean Thinking*, *6*, 21–46.

Kim, D., Uma, R. N., Abay, B. H., Wu, W., Wang, W., & Tokuta, A. O. (2014). Minimum latency multiple data muletrajectory planning in wireless sensor networks. *IEEE Transactions on Mobile Computing*, *13*, 838–851.

Kim, D., Xue, L., Li, D., Zhu, Y., Wang, W., & Tokuta, A. O. (2017). On theoretical trajectory planning of multiple drones to minimize latency in search-and-reconnaissance operations. *IEEE Transactions on Mobile Computing*, *16*, 3156–3166.

Kramer, R., Subramanian, A., Vidal, T., & Cabral, L. (2015). A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, *243*, 523–539.

Kucukoglu, I., Dewil, R., & Cattrysse, D. (2021). The electric vehicle routing problem and its variations: A literature review. *Computers & Industrial Engineering*, *161*, Article 107650.

Layeb, A., Ammi, M., & Chikhi, S. (2013). A grasp algorithm based on new randomized heuristic for vehicle routin problem. *Journal of Computing and Information Technology*, *1*, 35–46.

Liang, J., Huang, X., & Zhang, Z. (2019). Approximation algorithms for distance constraint sweep coverage with base stations. *Journal of Combinatorial Optimization*, *37*, 1111–1125.

Liu, Z., hao Wang, Z., Leo, D., Liu, X., & Zhao, H. (2017). QUADO: an autonomous recharge system for quadcopter. In *Int. conf. on cybernetics and intelligent systems (CIS) and conf. on robotics, automation and mechatronics* (pp. 7–12).

Liu, Z.-N., Liu, X.-Q., Yang, L.-J., Leo, D., & Zhao, H. (2018). An autonomous dock and battery swapping system for multirotor uav. Unpublished, https://www.researchgate.net/publication/325077351, 10.

Luo, C., Miao, W., Ullah, H., McClean, S., Parr, G., & Min, G. (2019). Unmanned aerial vehicles for disaster management. In *Geological disaster monitoring based on sensor networks* (pp. 83–107). Singapore: Springer.

Mancini, S. (2016). A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research Part C (Emerging Technologies)*, *70*, 100–112.

Mancini, S., Gansterer, M., & Hartl, R. (2021). The collaborative consistent vehicle routing problem with workload balance. *European Journal of Operational Research*, *293*(3), 955–965.

Masmoudi, M., Hosny, M., Braekers, K., & Dammak, A. (2016). Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, *96*, 60–80.

Mingozzi, A., Roberti, R., & Toth, P. (2013). An exact algorithm for the multitrip vehicle routing problem. *INFORMS Journal on Computing*, *25*, 193–207.

Nadi, A., & Edrisi, A. (2017). Adaptive multi-agent relief assessment and emergency response. *International Journal of Disaster Risk Reduction*, *24*, 12–23.

Nagarajan, V., & Ravi, R. (2012). Approximation algorithms for distance constrained vehicle routing problems. *Networks*, *59*, 209–214.

Petch, R., & Salhi, S. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematic*, *133*, 69–92.

Poikonen, S., Wang, X., & Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, *70*, 34–43.

Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, *23*, 229–235.

Resende, M., & Ribeiro, C. (2010). Handbook of methaueristics. In *Chapter greedy randomized adaptive search procedures: advances, hybridizations and applications*. Springer.

Sahin, M., & Yaman, H. (2022). A branch and price algorithm for the heterogeneous fleet multi-depot multi-trip vehicle routing problem with time windows. *Transportation Science*.

Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., & Guizani, M. (2019). Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *IEEE Access*, *7*, 48572–48634. http://dx.doi.org/10.1109/ACCESS.2019.2909530.

Sharifi, F., Zhang, Y., & Aghdam, A. G. (2014). Forest fire monitoring and detection using a network of autonomous vehicles. In *Proceedings of international conference on intelligent unmanned systems: vol. 10*.

Taillard, É. D., Laporte, G., & Gendreau, M. (1996). Vehicle routeing with multiple use of vehicles. *Journal of the Operational research society*, *47*, 1065–1070.

Talarico, L., Meisel, F., & Sorensen, K. (2015). Ambulance routing for disaster response with patient groups. *Computers & Operations Research*, *56*, 120–133.

Tarantilis, C., Zachariadis, E. E., & Kiranoudis, C. (2008). A hybrid guided local search for the vehicle routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing*, *20*, 154–168.

Thiels, C. A., Aho, J. M., Zietlow, S. P., & Jenkins, D. H. (2015). Use of unmanned aerial vehicles for medical product transport. *Air Medical Journal*, *34*, 104–108.

Valavanis, K. P., & Vachtsevanos, G. J. (2014). *Handbook of unmanned aerial vehicles*. Springer Publishing Company, Incorporated.

Vidal, T., Crainic, T., Gendreau, M., Lahrichi, N., & Rei, W. (2011). A hybrid genetic algorithm for multi depot and periodic vehicle routing problems. *Operations Research*, *60*, 611–624.

Yu, W., & Liu, Z. (2016). Improved approximation algorithms for some min–max and minimum cycle cover problems. *Theoretical Computer Science*, *654*, 45–58.

Zhan, C., Zeng, Y., & Zhang, R. (2018). Energy-efficient data collection in UAV enabled wireless sensor network. *IEEE Wireless Communications Letters*, *7*, 328–331.

Zhang, Q., Wang, Z., Huang, M., Yu, Y., & Shu-Cherng, F. (2022). Heterogeneous multi-depot collaborative vehicle routing problem. *Transportation Research, Part B (Methodological)*, *160*, 1–20.

Zhen, L., Ma, C., Wang, K., Xiao, L., & Zhang, W. (2020). Multi-depot multi-trip vehicle routing problem with time windows and release dates. *Transportation Research Part E: Logistics and Transportation Review*, *135*, Article 101866.