A Matheuristic for Multi-Depot Multi-Trip Vehicle Routing Problems

Tiziana Calamoneri[1], Federico Corò[2](✉), and Simona Mancini[3,4]

[1] Sapienza University of Rome, Rome, Italy
calamo@di.uniroma1.it

[2] Missouri University of Science and Technology, Rolla, USA
federico.coro@mst.edu

[3] University of Palermo, Palermo, Italy
simona.mancini@unipa.it

[4] University of Klagenfurt, Klagenfurt am Wörthersee, Austria
simona.mancini@aau.at

Abstract. Starting from a real-life application, in this short paper, we propose the original *Multi-Depot Multi-Trip Vehicle Routing Problem with Total Completion Times minimization (MDMT-VRP-TCT)*. For it, we propose a mathematical formulation as a MILP, design a matheuristic framework to quickly solve it, and experimentally test its performance.

It is worth noting that this problem is original as in the literature its characteristics (*i.e.*, multi-depot, multi-trip and total completion time) can be found separately, but never all together. Moreover, regardless of the application, our solution works in any case in which a multi-depot multi-trip vehicle routing problem must be solved.

Keywords: Matheuristic · Multi-trip · Multi-depot · Vehicle routing problems · Total completion time minimization

1 Introduction

In this short paper, we study the *Multi-Depot Multi-Trip Vehicle Routing Problem with Total Completion Times minimization (MDMT-VRP-TCT)*. This problem arises from a Search & Rescue application: immediately after a natural disaster, a fleet of unmanned aerial vehicles (UAVs) helps rescue teams to individuate people needing help inside an affected area. In this context, typically diverse civil defence rescue teams rush from the vicinity to the most affected area, so they give rise to multiple depots. Moreover, UAVs return to depots to substitute their batteries and leave for a new tour, so introducing a multi-trip scenario. Finally, to save as many lives as possible, the most important goal is to get the job done in the shortest possible time, so we aim at minimizing the total completion time.

The resulting optimization problem is original, as in the literature these three characteristics can be found separately, but never all together. Indeed, many problems having similarities with ours can be found, but they also have essential differences with respect to ours.

In this short paper we only refer to [2] as a survey paper on Multi-Trip Vehicle Routing Problems (MTVRP) (where there is a single depot), to [3] for the description of Multiple Traveling Repairperson Problem (mTRP) (that could appear similar to ours but the latency is minimized instead of the completion time), to [1] for the Multiple Traveling Salesperson Problem (mTSP) (where there are no battery constraints *i.e.*, a single trip for each vehicle), and to [4] for a description of the Rooted Min-Max Cycle Cover Problem (RMMCCP) (with single depot).

The rest of this short paper is organized as follows: in Sect. 2, we model MDMT-VRP-TCT as a MILP; in Sect. 3 we propose a matheuristic framework to face reasonably large instances and, in Sect. 4, we experimentally test it.

2 Mathematical Formulation

In this section, we present a mathematical formulation for MDMT-VRP-TCT.

Assume to have an area of interest (*e.g.* the one affected by a natural disaster) with a set I of *target nodes* to monitor (*e.g.* all the damaged buildings). Around the area, there is a set D of *depots* where a set U of vehicles start from (*e.g.* the places where some rescue teams settle down their bases, each one with a sub-fleet of UAVs); in general, each vehicle u is characterized by a different *budget* b_u and then it has to come back to the depot it is uniquely associated to (*e.g.* each UAV has a battery; when it runs down, it is necessary to substitute it with a charged one, and this can be done only at its own depot).

The *traveling distance* between each pair of nodes $i, j \in I \cup D$ is known. Each node $i \in I$, has an associated *service time* s_i (*e.g.* the needed time to overfly it).

A *sequence* is any ordered set k of target nodes; the *duration* of sequence k is computed as the sum of all traveling distances between consecutive target nodes in k plus the service times of all the target nodes in k.

The aim of our problem consists in assigning to each vehicle $u \in D$ an ordered set of sequences such that, from its depot, u is able to reach the first target of any of the sequences assigned to it, serve all its target nodes, come back to o_u, and start again. A sequence k with the addition of the depot of u is called a *trip* and its duration, t_{ku} is given by the duration of k plus the traveling distances between the depot associated to u and the two extremes of k.

A sequence k is *compatible* with a vehicle u if its duration is upper bounded by b_u. A compatibility index, Φ_{ku} is defined equal to 1 if sequence k is compatible with vehicle u and equal 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 it is compatible with at least one vehicle. Only feasible sequences are considered.

A *solution* for our problem consists in selecting a set of sequences K whose union covers I and in assigning them to compatible vehicles. We define the *total completion time of a solution* the maximum over all times required by each UAV to fly over all the trips assigned to it by that solution.

Then, we introduce the following decision variables:

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

The mixed integer programming formulation is reported in the following

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

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

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

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

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

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

The objective function consisting into the minimization of the total completion time, as reported in (of). Constraints (C1) ensure that each target is covered by exactly 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 (C3). The total completion time must be larger than the cumulative working time of each vehicle, as stated in (C4). This formulation distinguishes from the trip based standard one in the objective function.

We conclude this section highlighting that the novelty of our approach lies in considering sequences that can be assigned to vehicles located in different depots (in fact to all vehicles whose depot position makes them compatible with them) instead of constructing closed trips (as for example in [5]), that are inherently partitioned among the vehicles. Moreover, regardless of the application, our modeling approach works in any case in which a multi-depot multi-trip vehicle routing problem must be solved, whichever is the objective function to be optimized. Therefore, it can be applied also in cases in which the goal is to minimize the total covered distance, as common in logistics applications.

3 A Model Based Matheuristic Framework

The main idea under the above presented mathematical model consists into generating all possible feasible sequences and associating them to the set of their compatible vehicles. When the number of feasible trips is too large to be handled, the mathematical model becomes intractable. If, for instance, target nodes are very near among each other, or batteries are very large, so that several target nodes can be visited in a single sequence, even small instances may become difficult to handle exactly.

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

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

A sequence k is *dominated (strictly dominated)* by another one, k', if they have the same extremes and contain exactly the same target nodes (possibly in a different order), but k' has a lower or equal (strictly lower) duration than k.

We initially generate all the sequences containing only one target node and directly insert them in the set of sequences to be passed to the model, $\tilde{K}$. We also insert them in a temporary set of sequences K^{tmp} which contains sequences to be expanded. All the sequences included in K^{tmp} are then processed. N_c child sequences are generated from each sequence k adding after the last target node in the sequence, l_k, the j^{th} nearest node to l_k among those not already included in k, with j varying in $\{1, N_c\}$. For each child sequence k^c, we apply a first feasibility check: if the duration sequence k^c is larger than the maximum autonomy of a vehicle, $B_{max} = \max_{u \in U} b_u$, then the sequence is immediately discarded. Otherwise, we pass the corresponding trip to a second feasibility check, in order to verify that the sequence is neither strictly dominated by nor it strictly dominates another sequence already belonging to $\tilde{K}$. If a domination occurs, the dominated sequence is discarded otherwise it is kept in $\tilde{K}$. At this point, we set, for all the vehicles u compatible with k, $\Phi_{ku} = 1$, and for all the others $\Phi_{ku} = 0$.

Once all the child sequences of a sequence k have been analyzed, k is removed from K^{tmp}. The procedure terminates when K^{tmp} is empty or when a maximum allowed number of sequences, K_{max} have been added to $\tilde{k}$.

K_{max} is a parameter of the algorithm and it plays a crucial role in the performance of the algorithm. A larger value of K_{max} would yield to a better global solution but would increase the computational time required by the heuristic. To obtain an effective and efficient algorithm, this parameter must be carefully tuned in order to achieve a good balance between solutions quality and computational times. The maximum number of children generated by each trip, N_c, also plays an important role. The larger the value of N_c, the larger the number of sequences containing a specific number of target nodes. Note that keeping fixed the value of the maximum number of sequences to generate 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 only nodes which are very close to each others are visited sequentially, and this would imply that isolated targets would appear in very few sequences. Instead, with very large values of N_c, even targets which are not very close to each others can be visited, but in this case each sequence have several children, and so the maximum allowed number of sequence is reached already considering sequences containing a small number of targets, longer sequences are not generated, and this could negatively affect the solution quality.

After the sequence generation process is finished, the set of sequences $\tilde{K}$ is given in input to the mathematical model ((of)–(C4)).

4 Computational Results and Discussion

In this section, we study the performance of our matheuristic, comparing it with the exact model on small instances) and varying some parameters of the problem.

In this short paper we selected only some experiments, shown in Fig. 1.

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 to note that when n is small ($n \leq 30$), there are results corresponding to the model; when $n = 40$, the model terminates only in 11 cases out of 20, and computational times varied between 101.38 and 30559.5 s; probably the corresponding instances are particularly easy to solve (*e.g.*, without clustered target nodes) and for this reason we decided not to report the average value of the optimal solution. Instead, when $n = 50$, the model is not able to terminate in any case.

The experiments perfectly confirm the expectations. Indeed:

- The first three charts in the first row show the percentage gap from the optimum completion time, that is the main objective function of our problem; it is clear that it tends to 0 as N_c grows up and, when $N_c = 6$, it is very close to 0, showing that our matheuristic works very well. We can also observe that also with $N_c = 3$ the gap is very small for instances with 10 nodes, while it tends to increase for larger instances. The last two charts of the first row, instead, show the percentage gap w.r.t. the case $N_c = 3$; since large values of N_c lead to better solutions, clearly, these gaps are negative.
- The second row of charts corresponds to the number of trips generated in order to individuate the solution. As expected, the matheuristic dramatically decreases the number of trips, that is higher and higher when N_c increases but anyway reasonable. Just this reduction makes the matheuristic tractable even for large instances.
- The third row of charts corresponds to the time necessary for running the model and the matheuristic. Clearly, the computational time of the model is much higher and, for what concerns the matheuristic, it grows up when N_c is increased.

The novelty of the approach, consists into generating (open) sequences of nodes, that can be assigned to different vehicles at different costs, instead of generating complete routes including the depot. This way, the problem can be

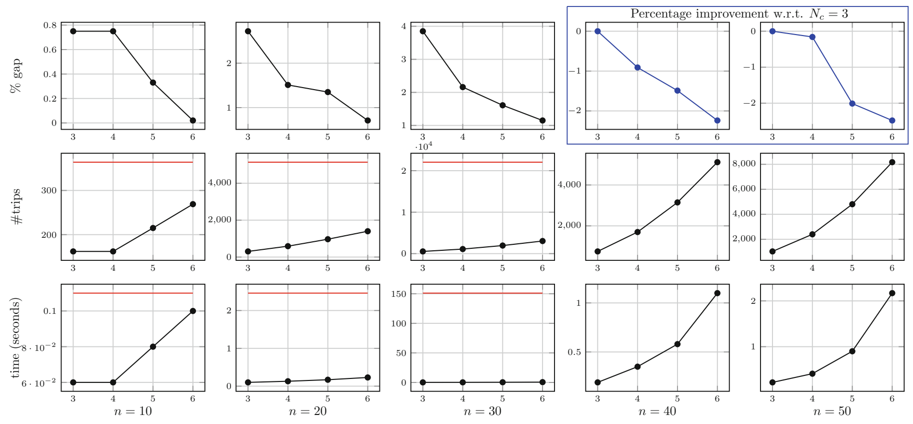

Fig. 1. Experimental results. On the x axis, 3,4,5,6 represent the used values of N_c; the red lines represent the benchmark values achieved by the model. (Color figure online)

modeled as a multiple choice knapsack, with knapsack-dependent items weight and maximum knapsack occupancy minimization. Such approach is not only valid for this specific problem, but can be used for a wide class of multi-depot multi-trip problems, including those having different objective functions, such as the *classical* total travel distance minimization, the minimization of the number of vehicles used, or the minimization of total cost given by vehicles purchasing costs plus travel costs. Furthermore, this matheuristic framework can be used whichever is the method exploited to generate promising sequences. In particular, it can be combined with well known solutions generation algorithms such as the Greedy randomized adaptive search (GRASP). However, we believe our method is more suitable for problems with heterogeneous fleet, since we generate sequences of increasing length, among which, the smaller ones are compatible also with vehicles with a limited autonomy (or capacity). Conversely, GRASP is designed for problems with homogeneous fleet and tend to generate sequences which exploit the whole capacity/autonomy of the vehicle.

References

1. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega **34**(3), 209–219 (2006)
2. Cattaruzza, D., Absi, N., Feillet, D.: Vehicle routing problems with multiple trips. 4OR **14**(3), 223–259 (2016). https://doi.org/10.1007/s10288-016-0306-2
3. Méndez-Díaz, I., Zabala, P., Lucena, A.: A new formulation for the traveling deliveryman problem. Discret. Appl. Math. **156**(17), 3223–3237 (2008)
4. Nagarajan, V., Ravi, R.: Approximation algorithms for distance constrained vehicle routing problems. Networks **59**, 209–214 (2012)
5. Paradiso, R., Roberti, R., Laganà, D., Dullaert, W.: An exact solution framework for multitrip vehicle-routing problems with time windows. Oper. Res. **68**(1), 180–198 (2020)