A Study on the Traveling Salesman Problem with a Drone[⋆]

Ziye Tang[1], Willem-Jan van Hoeve[1], and Paul Shaw[2]

[1] Tepper School of Business, Carnegie Mellon University, USA
ziyet@andrew.cmu.edu,vanhoeve@andrew.cmu.edu
[2] IBM, France paul.shaw@fr.ibm.com

Abstract. A promising new model for future logistics networks involves the collaboration between traditional trucks and modern drones. The drone can pick up packages from the truck and deliver them by air while the truck is serving other customers. The operational challenge combines the allocation of delivery locations to either the truck or the drone, and the coordinated routing of the truck and the drone. In this work, we consider the scenario of a single truck and one drone, with the objective to minimize the completion time (or makespan). As our first contribution, we prove that this problem is strongly NP-hard, even in the restricted case when drone deliveries need to be optimally integrated in a given truck route. We then present a constraint programming formulation that compactly represents the operational constraints between the truck and the drone. Our computational experiments show that solving the CP model to optimality is significantly faster than the state-of-the-art exact algorithm. For larger instances up to 60 delivery locations, our CP-based heuristic algorithm is competitive with a state-of-the-art heuristic method in terms of the solution quality.

1 Introduction

Vehicle routing problems have become increasingly important with the evolution of online shopping and fulfillment and a variety of delivery services. The use of unmanned aerial vehicles, or drones, for this purpose is actively explored by industry [12]. A common model is to equip a delivery truck with one or more drones to deliver packages in parallel to the truck [15]. Unlike the traditional setting where a fleet of vehicles have little operational constraints to each other, the drone operation is highly constrained to the truck operation because it needs to pick up packages for delivery from a truck. As a result the completion time also depends on the waiting time incurred due to the synchronization between the truck and the drone.

In this paper we study the design of optimal joint truck and drone routes under this scenario. We consider the elementary case where only one truck and one drone is available. Given a set of customers to be served either by a truck

[⋆] This work was supported by Office of Naval Research grant N00014-18-1-2129.

or a drone, our objective is to minimize the completion time of the entire delivery task, *i.e.* the total time it takes to serve all customers. For operational simplicity, we assume the drone can only be dispatched at a customer location and the service time at each location is instant. In a feasible solution, the truck route forms a tour which starts from and returns to the depot with a subset of customers served along the tour. Each remaining customer is served by the drone which is dispatched from a customer location and returns to a (possibly different) location on the truck tour. We follow Agatz *et al.* [1] and call this problem the *traveling salesman problem with a drone* (TSP-D).

Contributions. Our first contribution is a proof that the TSP-D is strongly NP-hard even in the case when a truck route is given and we need to optimally integrate the remaining drone visits. Our second contribution is a new constraint programming (CP) formulation that relies on representing the TSP-D as a scheduling problem. We show experimentally that our CP approach outperforms the best exact method from the literature, and is competitive with a state-of-the art heuristic method in terms of solution quality.

2 Related Work

The hybrid truck and drone model was first studied by Murray *et al.* [11]. Agatz *et al.* [1] propose an exponential-sized integer programming model. Ha *et al.* [6] consider a variant where the objective is to minimize operational costs including total transportation cost and the cost incurred by vehicles' waiting time. Ham [7] considers a different integrated model where after one delivery task, the drone may return to the depot or fly to another customer to pick up a return order from a customer, with the truck traveling separately along a cycle. Bouman *et al.* [2] introduce a dynamic programming (DP) formulation of the TSP-D and solve it with A^* search. Yurek and Ozmutlu [17] propose a decomposition method; in the first stage, the truck nodes and the truck routes are generated and determined and the second stage solves a mixed-integer program to determine the optimal drone schedule. Lastly, Poikonen *et al.* [13] develop a specialized branch-and-bound procedure, which includes boosted lower bound heuristics to further speed up the solving process. Their method assumes insertion of a customer node into a sequence of nodes will not increase the optimal cost, which does not hold when the drone has a finite flight range. Other heuristic algorithms have also been proposed in the literature, see *e.g.* [1,3,4,5,11]. The best exact method for the TSP-D is the DP approach in [2], which can optimally solve instances with up to 15 locations in reasonable time.

The first theoretical study was performed by Wang *et al.* [16], who consider the more general vehicle routing problem with multiple trucks and drones. They study the maximum savings that can be obtained from using drones compared to truck-only deliveries (*i.e.* TSP cost) and derive several tight theoretical bounds under different truck and drone configurations. Poikonen *et al.* [14] extend [16] to different cases by incorporating cost, limited battery life and different metrics respectively.

3 Problem Definition

We are given an undirected graph $G = (V, E)$ with $V = C \cup \{r\}$ where r is the depot and C is the set of customers to be served by the truck or the drone. Let $n = |C|$. The travel time between a pair of nodes (i, j) is given by metric $w(i, j)$. $\rho \geq 0$ is the ratio between the truck's and the drone's travel time per unit distance. ρ is also called the *speed differential*. Every customer demands one parcel, which can be delivered by either a truck or a drone. A drone can only deliver one parcel at the time. We make the following assumptions about the behavior of the truck and the drone:

(a) The truck can dispatch and pick up a drone only at the depot or a customer location. The truck can continue serving customers after a drone is dispatched and reconnect with the drone at a possibly different node.
(b) The vehicle (truck or drone) that first arrives at the reconnection node has to wait for the other one, which we call *synchronization*.
(c) Upon returning to the truck, the time required to prepare the drone for another launch is negligible.

Our objective is to minimize the completion time, *i.e.* from the time the truck is dispatched from the depot with the drone to the time when the truck and the drone returns to the depot.

Notation. In a feasible solution to TSP-D, denote V_d as the set of nodes visited by the drone only. Denote $V_t := V \backslash V_d$ as the set of nodes including the depot visited by the truck either with or without the drone atop the truck. By a slight abuse of notation, we also call V_d the set of *drone nodes* and V_t the set of *truck nodes*. For each $i \in V_d$, let $p(i)$ be the dispatch node where the drone is dispatched right before visiting i, $q(i)$ be the pick up node where the drone returns immediately after visiting i. Let E_t be the set of edges in the truck tour. For $i, j \in V_t$, let T_{ij} denote the path induced by E_t and $w(T_{ij}) = \sum_{e \in T_{ij}} w_e$. Consider a partial drone schedule where the drone is dispatched from the truck at node j, visits node i and meets up with the truck at node k (we allow $j = k$). We call this partial drone schedule a *drone activity* and use a shorthand notation $j \to i \to k$ to represent this activity.

4 Computational Complexity

Solving the TSP-D to proven optimality is highly challenging, as witnessed by the performance of the best exact methods—they scale up to about 15 locations only. While the TSP-D is known to be NP-hard due to a reduction from the TSP, we aim to provide more insight in the computational difficulty by considering a restricted version, which we call the drone routing subproblem (DRS). We next prove strong NP-hardness of this restricted version.

We associate the drone activity $j \to i \to k$ with a cost c_{ijk} defined as

$$c_{ijk} = \max\{0, w(j, i) + w(i, k) - \rho w(T_{jk})\} \tag{1}$$

This is the marginal time a drone activity adds to the truck tour. Given V_t, V_d and E_t, we define a set of drone activities to be *feasible* if (1) each drone node in V_d appears in exactly one drone activity and (2) any pair of activities do not overlap in time.

We now define the drone routing subproblem as follows:

Definition 1 (Drone routing subproblem). *Given V_t, V_d, E_t. The drone routing subproblem is to find a feasible set of drone activities with minimum total drone activity cost.*

We show that DRS is strongly NP-hard by a reduction from 3-partition.

Definition 2 (3-Partition). *Given positive integers m, B and $3m$ positive integers $x_1, \ldots, x_{3m}$ satisfying $\sum_{q=1}^{3m} x_q = mB$ and $\frac{B}{4} < x_q < \frac{B}{2}$ for $q = 1, \ldots, 3m$. Does there exist a partition of the set $Y = \{1, \ldots, 3m\}$ into m disjoint subsets $Y_1, \ldots, Y_m$ such that $\sum_{q \in Y_i} x_q = B$ for $i = 1, \ldots, m$?*

Theorem 1. *The drone routing subproblem is strongly NP-hard.*

Proof. We prove the theorem for $\rho = 1$. We give a pseudo-polynomial time reduction from 3-partition. Given an instance of 3-partition as in Definition 2, we construct a graph with $m(B+1)$ truck nodes and $4m$ drone nodes. The truck route connects m paths P_i, each having $B+1$ nodes and B unit edges. Edges that connect two paths are assigned $\epsilon = \frac{1}{2m}$. Direct all edges in the cycle counterclockwise. The tail of a directed path P_i is defined as the tail of the first arc in P_i. We similarly define the head of P_i. The drone nodes are partitioned into two disjoint sets A and B. A has $3m$ nodes $v_1, \ldots, v_{3m}$. For $i = 1, \ldots, 3m$, v_i is connected to each node on the cycle via an edge of weight $\frac{x_i}{2}$. B contains m dummy nodes $u_1, \ldots, u_m$. For $i = 1, \ldots, m$, each u_i is connected to the head of P_i and tail of P_{i+1} via two edges of weight $\frac{\epsilon}{2}$ (we assume $P_{m+1} = P_1$). Other edges connected to u_i are assigned a unit weight so metric inequality holds. Below we show Lemma 1, from which the theorem follows. $\square$

Lemma 1. *There exists a 3-partition if and only if there exists a feasible solution to the above DRS instance of zero total cost.*

Proof. 'if': connect each dummy node u_i to the head of P_i and tail of P_{i+1}. Without loss of generality assume the feasible partition is $\{x_{3k+1}, x_{3k+2}, x_{3k+3}\}$ for $k = 0, \ldots, m-1$. Then $v_{3k+1}, v_{3k+2}, v_{3k+3}$ are connected to path P_k in the following way: v_{3k+1} is connected to the first node and $(x_{3k+1}+1)$-th node on the path, v_{3k+2} is connected to $(x_{3k+1}+1)$-th and $(x_{3k+1}+x_{3k+2}+1)$-th nodes, v_{3k+3} is connected to $(x_{3k+1}+x_{3k+2}+1)$-th and x_{B+1}-th nodes. It is easy to check that the total cost is zero.

'only if': we claim each dummy node u_i in any solution with zero total cost must be connected to the head of P_i and tail of P_{i+1}: suppose not, note for any $t \neq i$, u_i cannot be connected to the head of P_t and tail of P_{t+1} since otherwise such a drone activity has non-zero cost. As a result any drone activity which

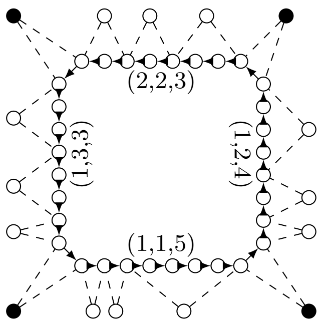

Fig. 1. An example of the reduction, with $m = 4$, $B = 7$ and feasible partitions $(1, 1, 5)$, $(1, 2, 4)$, $(1, 3, 3)$, $(2, 2, 3)$. Drone activities are shown as dashed lines and dummy nodes are marked as solid. While each drone node has the same distance to any node on the truck cycle, we put the drone nodes outside the cycle for visualization purposes.

visits u_i covers at least a unit-length edge on the cycle. Therefore after visiting u_i, remaining edges on the cycle have at most $mB - 1 + m\epsilon = mB - \frac{1}{2} < mB$ length to use for visiting the remaining drone nodes. Notice each visit of a node $v_l \in A$ must cover a path at least x_l to make the drone activity cost zero and each visit must cover non-overlapping path on the cycle, which is a contradiction to the fact that the remaining edge length on the cycle is less than mB. Therefore we've shown the claim. The result follows by reversing the steps in the 'if' part to partition drone nodes into m sets where each set contains 3 nodes that are visited by using edges in the same path. □

5 Constraint Programming Formulation

An essential feature of a truck-drone schedule is the synchronization between truck and drone operations. This poses a significant challenge to construct a MIP model with a tight linear relaxation. Below we explain how to construct a compact CP with $O(n^2)$ variables and constraints, using the constraint-based scheduling formalism introduced in [8,9,10]. The CP solver IBM ILOG CP Optimizer [10] provides an expressive modeling language based on the notion of interval variables representing the execution of an activity. Its domain encodes the presence status (Boolean) (true if the activity is executed). When a is present, it is represented by variables $s(a)$ for its start time, $e(a)$ for its end time, and $d(a)$ for its duration, obeying the relationship $d(a) = e(a) - s(a)$. On the other hand, an absent interval variable is not considered by any constraint or expression on interval variables in which it is involved. An activity can be forced to present or declared 'optional', *i.e.* its presence status can be either true or false to be decided by the solver. Below we assume all interval variables are optional unless stated otherwise.

Recall the number of nodes including the depot is $n + 1$. Denote both node 0 and n as the depot (leaving and returning). For each node i define three in-

```
1   minimize
2      endOf(tVisit[n])
3   subject to {
4      forall (i in 0...n) setPresent(visit[i])
5      setPresent(tVisit[0])
6      setPresent(tVisit[n+1])
7      first(tVisit, tVisit[0])
8      last(tVisit, tVisit[n])
9      no_overlap(tVisit, w)
10     no_overlap(dVisit)
11     forall(i in 0...n) {
12        alternative(visit[i], [tVisit[i], dVisit[i]])
13        alternative(dVisit_before[i], [all (j in 0...n) tdVisit[i][j]])
14        alternative(dVisit_after[i], [all (j in 0...n) dtVisit[i][j]])
15        span(dVisit[i], [dVisit_before[i], dVisit_after[i]])
16        end_at_start(dVisit_before[i], dVisit_after[i])
17        if_then(presence_of(dVisit[i]), presence_of(dVisit_before[i]) & presence_of(dVisit_after[i]))
18     forall(i, j in 0...n) {
19        if_then(presence_of(tdVisit[i][j]), presence_of(tVisit[j]))
20        if_then(presence_of(dtVisit[i][j]), presence_of(tVisit[j]))
21        start_before_start(tVisit[j], tdVisit[i][j])
22        start_before_end(tdVisit[i][j], tVisit[j])
23        end_before_end(dtVisit[i][j], tVisit[j])
```

Fig. 2. A compact constraint programming formulation for TSP-D. Note we can enforce finite drone range by adding an upper bound on the duration of each `dVisit[i]`.

terval variables: `visit[i]` (forced to be present), `dVisit[i]` and `tVisit[i]`. Each `dVisit[i]` represents the time period from the drone just leaving for node i to the drone arriving at the first truck node after serving i. Note we can enforce finite drone range by adding an upper bound on the duration of each `dVisit[i]`. Furthermore, for each node i, we create two interval variables `dVisit_before[i]` and `dVisit_after[i]` which represent splitting `dVisit[i]` by the time point of the drone visiting i. For each pair (i, j), we define two interval variables `tdVisit[i][j]` and `dtVisit[i][j]`, where the former represents the drone leaving from truck node j to drone node i and the latter represents the drone leaving from drone node i to truck node j. Each `tdVisit[i][j]` is lower bounded by the drone travel time from j to i and similarly for `dtVisit[i][j]`. As an example, an activity $i \to j \to k$ is composed of `tdVisit[i][j]` and `dtVisit[i][k]`, which are constrained to be equivalent to `dVisit_before[i]` and `dVisit_after[i]` respectively. The complete model is presented in pseudocode in Figure 2.

Lines 7-8 require the truck tour to start and end at the depot. The next constraint requires that for each pair (i, j), their truck visits have to be at least w_{ij} apart if both of them are served by the truck. Similarly for line 10. The remaining constraints enforce logical constraints between different sets of interval variables. For example, `alternative`(*interval*, *array*) creates an alternative constraint between interval variable *interval* and the set of interval variables in *array*. If *interval* is present, then one and only one of the intervals in *array* will be selected by the alternative constraint to be present and the start and end values of *interval* will be the same as the one of the selected intervals. Line 21-23

Size	10	11	12	13	14	15	16	17	18
CP	6.79	5.71	16.66	15.66	50.83	120.59	216.46	375.49	564.22
DP	1.00	4.00	12.00	56.00	306.00	1568.00	9508.00	–	–

a. Runtime comparison (s) of CP and DP (exact).

Size	10	20	30	40	50	60	70	80	90	100	200
CP	116.60	136.64	160.12	198.88	237.4	276.96	316.20	407.36	515.80	679.64	–
BAB	149.53	171.64	200.95	226.15	241.36	267.54	283.30	299.09	322.37	337.91	465.63

b. Solution value comparison of CP and BAB (heuristic).

Table 1. Comparison of our constraint programming (CP) approach with (a) the exact dynamic programming (DP) method of Bouman *et. al.* [2] in terms of runtime (s), and (b) the heuristic branch-and-bound method (BAB) of Poikonen *et al.* [13] in terms of solution quality (average objective value).

implements the synchronization constraint between the truck and the drone. We refer the reader to the manual [8] for the definition and usage of each constraint.

6 Computational Experiments

We implemented and solved our CP model with CP Optimizer version 12.8.0, using the Python interface DOcplex. Our experiments are run on a 2.2GHz Intel Core i7 quad-core machine with 16GB RAM. We compared our approach to the two best approaches from the literature: the exact dynamic programming (DP) algorithm in [2], and the branch-and-bound algorithm from [13]. The implementation of the latter relies on the assumption that the drone has a finite range, for which the method is not guaranteed to provide optimal solutions.

We first present the results on the exact comparison. Since benchmark instances used in [2] are not publicly available, we follow their approach to generate 10 uniform instances of each size. We use the same parameter $\rho = 2$ as speed differential. Table 1.a reports the average runtime (in seconds) of our approach (CP) and the reported runtime from [2] (DP). While DP can solve the smaller problems faster than CP, our approach scales more gracefully.

Table 1.b presents the comparison with the branch-and-bound method (BAB) of [13] in terms of solution quality. For this experiment, we apply a time limit of 10 minutes for each instance. As a benchmark, we use the same dataset as [13] (25 instances of each size). We use the same parameter values for the speed differential $\rho = 2$ and drone range $R = 20$. The table reports the mean objective value for each problem size. The results for BAB are the best solutions found among all branch-and-bound heuristic variants. We note that the runtime of the BAB approach is typically less than one minute. In comparison, we terminate our CP model at a time limit of 10 minutes. These results show that the time-limited CP approach can produce competitive solutions for instances of up to

60 locations but that the dedicated heuristic branch-and-bound outperforms CP on larger instances.

Acknowledgement

The first author would like to thank Thomas Bosman for helpful discussions on the complexity proof.

References

1. Agatz, N., Bouman, P., Schmidt, M.: Optimization approaches for the traveling salesman problem with drone. Transportation Science **52**(4), 965–981 (2018)
2. Bouman, P., Agatz, N., Schmidt, M.: Dynamic programming approaches for the traveling salesman problem with drone. Networks **72**(4), 528–542 (2018)
3. Daknama, R., Kraus, E.: Vehicle routing with drones. arXiv preprint arXiv:1705.06431 (2017)
4. Dorling, K., Heinrichs, J., Messier, G.G., Magierowski, S.: Vehicle routing problems for drone delivery. IEEE Transactions on Systems, Man, and Cybernetics: Systems **47**(1), 70–85 (2017)
5. Ha, Q.M., Deville, Y., Dung, P.Q., Hà, M.H.: Heuristic methods for the traveling salesman problem with drone. CoRR **abs/1509.08764** (2015)
6. Ha, Q.M., Deville, Y., Pham, Q.D., Hà, M.H.: On the min-cost traveling salesman problem with drone. Transportation Research Part C: Emerging Technologies **86**, 597–621 (2018)
7. Ham, A.M.: Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. Transportation Research Part C: Emerging Technologies **91**, 1–14 (2018)
8. IBM ILOG CPLEX: CP Optimizer 12.7 User's Manual (2017)
9. Laborie, P.: IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems. In: Proceedings of CPAIOR. LNCS, vol. 5547, pp. 148–162. Springer (2009)
10. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP optimizer for scheduling. Constraints **23**(2), 210–250 (2018)
11. Murray, C.C., Chu, A.G.: The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies **54**, 86–109 (2015)
12. Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E.: Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. Networks **72**(4), 411–458 (2018)
13. Poikonen, S., Golden, B., Wasil, E.: A Branch and Bound Approach to the TSP with Drone. INFORMS Journal on Computing (to appear)
14. Poikonen, S., Wang, X., Golden, B.: The vehicle routing problem with drones: Extended models and connections. Networks **70**(1), 34–43 (2017)
15. UPS: UPS Tests Residential Delivery Via Drone. Youtube (2017), `https://www.youtube.com/watch?v=xx9_6OyjJrQ`
16. Wang, X., Poikonen, S., Golden, B.: The vehicle routing problem with drones: Several worst-case results. Optimization Letters **11**(4), 679–697 (2017)
17. Yurek, E.E., Ozmutlu, H.C.: A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. Transportation Research Part C: Emerging Technologies **91**, 249–262 (2018)