Package Delivery with Trucks and Drones: The Horsefly Problem

Joseph. S. B. Mitchell
Stony Brook University

Gaurish Telang
Stony Brook University

I. Introduction

With recent advances in drone technology and their widespread availability, several companies, including Amazon and UPS, have been exploring the use of drones for delivering packages, potentially with higher throughput at lower cost. *"UPS has estimated that cutting off just one mile for the routes of each of the company's 66,000 delivery drivers would amount to $50 million in savings. For this reason, UPS is testing drone deliveries, using the top of its vans as a mini-helipad."* [1]. The video in [2] shows this concept in action.

We consider an optimal package delivery problem utilizing a truck and a drone, specified as follows. Let $S = \{p_1, \ldots, p_n\}$ be a given set of n customer sites in $\mathbb{R}^2$. A truck full of packages starts at point s (the depot) and has a delivery drone on its rooftop. The drone can carry one package at a time to a customer, then return to the truck for another package. The truck moves at maximum speed 1, while the drone flies at speed $\varphi \geq 1$; we refer to φ as the *speed ratio*. The goal is to compute a route for the truck and for the drone in order to complete the delivery of all n packages (and have the drone return back to the empty truck) as soon as possible – i.e., we seek to minimize *makespan* of the delivery process. We describe algorithms to address this optimal delivery problem. We give both provable approximation results (an $O(\log n)$-approximation algorithm) and some experimental results comparing some heuristics.

The above problem has been called the *Horsefly* problem [3]. It is easy to see that the Horsefly problem is NP-hard, from the Euclidean TSP. (If the speed ratio $\varphi = 1$, then an optimal solution is for the drone and truck to travel together from site to site on a TSP path.) Figure 1 shows an example of a routing for a truck and drone for 30 sites, with $\varphi = 5$.

The challenge in computing a solution to the Horsefly problem is to determine both the order of the sites in which they must be serviced by the drone and the rendezvous points of the truck and the drone (the points where the drone lands back on the truck to pick up the next package, and then depart to the next customer).

It is also assumed in the above model that the truck itself cannot make any deliveries: it is always the drone that must deliver a package to every site. (The variant in which the truck/driver also makes deliveries is also of interest but we defer that discussion to a full paper.)

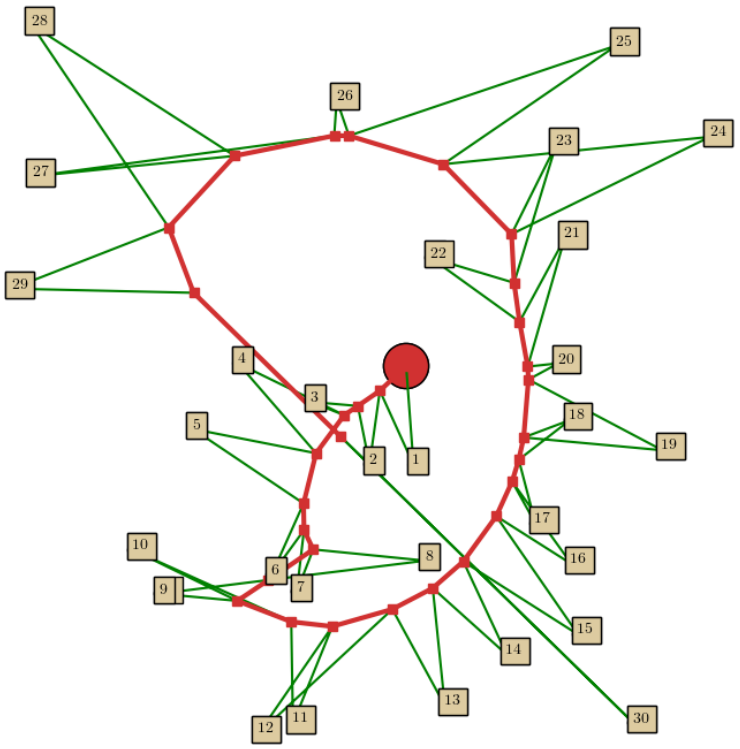


Fig. 1: Delivery with truck and drone, with speed-ratio $\varphi = 5$. The truck and drone travel along the red and green paths, respectively. The large red dot indicates the intial position s of the truck and drone.

II. An $O(\log n)$ approximation algorithm

The Horsefly problem in the plane has an $O(\log n)$-approximation algorithm. We briefly sketch the proof. The algorithm is based on computing, via dynamic programming, a least expensive solution of a particular structure: The truck traverses the edges of an orthogonal binary space partition (BSP), while the drone's routes are doubled line segments connecting each customer site p_i to the closest point on the boundary of the BSP face that contains it. The expense of a solution is a weighted sum of the edge lengths: φ times the total edge length of the BSP network (the truck network), plus 1 times the doubled segment lengths for the drone paths. We argue that an optimal solution to the Horsefly problem can be converted to a BSP-structured solution at a cost of a factor $O(\log n)$ in the makespan. From the optimal BSP-structured

network, we can extract optimal routes for the drone and truck, and a feasible delivery schedule, which must be within factor $O(\log n)$ of optimal.

III. Case: Delivery Order is Given

We consider first the case in which the order in which customer sites receive deliveries is given: $(p_1, p_2, \ldots, p_n)$.

We make some simple observations about the structure of an optimal solution:

(1) The truck route and the drone route are polygonal, with vertices at a set of departure/rendezvous points, $s_1 = s, s_2, s_3, \ldots, s_n$, where point s_i is a point along the truck route where the drone departs to deliver the package to customer p_i. (And, thus, s_i is also the point on the truck route where the drone returns to the truck after making the delivery to the customer at point p_{i-1}, for $i \geq 2$.)

(2) The truck and the drone move always at their maximum speeds (1 and φ, respectively).

Given the ordering of the sites, the problem of computing the optimal rendezvous points s_i can be formulated as a convex program, which can be solved by standard non-linear optimization solvers, such as the Sequential Least Squares Programming (SLSQP) solver from [7], which we use in our experiments. In order to speed up this computation, we formulate the problem of computing optimal tours in the L_1 (Manhattan) metric, which becomes a linear program.

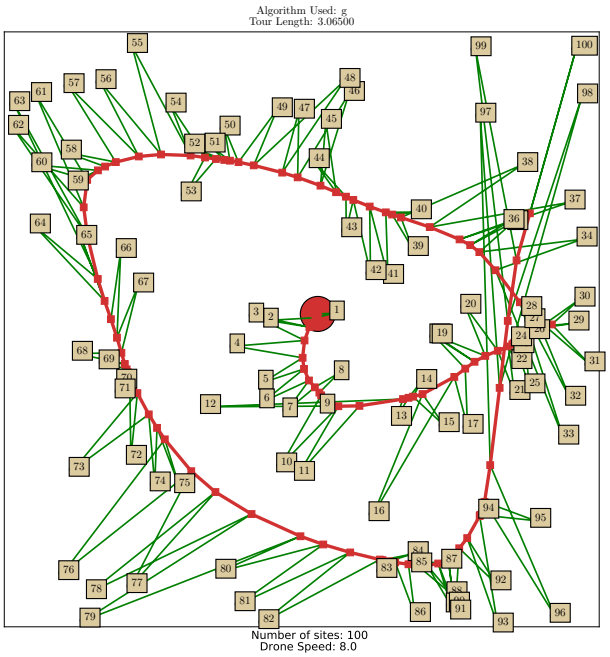
Figure 2 is an example of the output of using SLSQP versus LP obtained by the site-ordering given by the greedy heuristic for $\varphi = 8$. Note that the site *orderings* are the same for both heuristics. (The ordering was obtained using a greedy heuristic described in Section IV-A) The SLSQP based method took about 2 minutes to compute the tour whereas the LP based method computed the tour in less than a second. Both times include the time taken for computing the site ordering by the greedy heuristic

A. Comparing the exact and LP approximation ratio

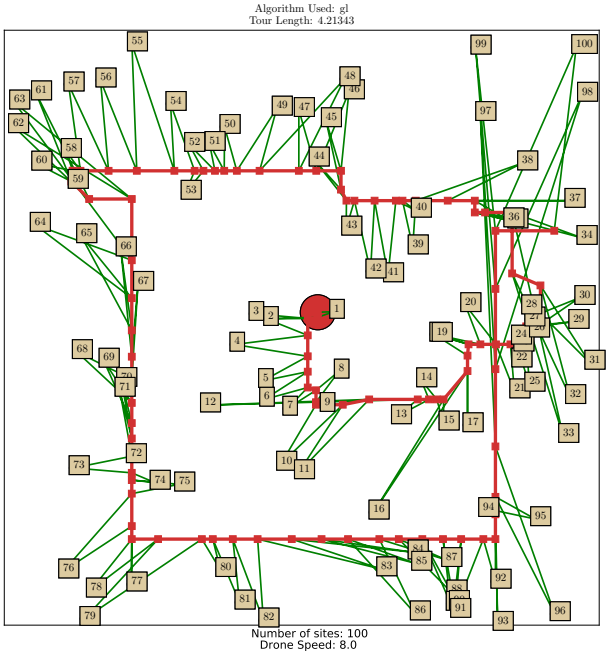
To test the approximation properties of the linear programming formulation, we compute the ratio of lengths of tours obtained via the LP heuristic and the exact SLSQP solver for different φ on 100 uniformly distributed sites in the unit square $[0, 1]^2$. We then repeat the experiment 40 times. The initial position of the horse and drone in all experiments was set to the middle of the square $(0.5, 0.5)$. We used the MOSEK [8] optimization package for solving the resulting linear programs.

The ratios of the tour lengths for each run and speed-ratio are plotted in the figure below, for various values of n (horizontal axis).

From Figure 3 it appears that the ratios of the tour-lengths is bounded for a fixed-speed φ. Further the worst-case ratio seems to increase slowly as a function of φ.



(a) Tour using SLSQP on site ordering returned by Greedy heuristic. Tour length 3.06500



(b) Tour using LP on site ordering returned by Greedy heuristic: Tour Length 4.21343

Fig. 2: An example of the tours computed via SLSQP and LP methods. Tour ordering computed via a greedy heuristic described in Section IV-A

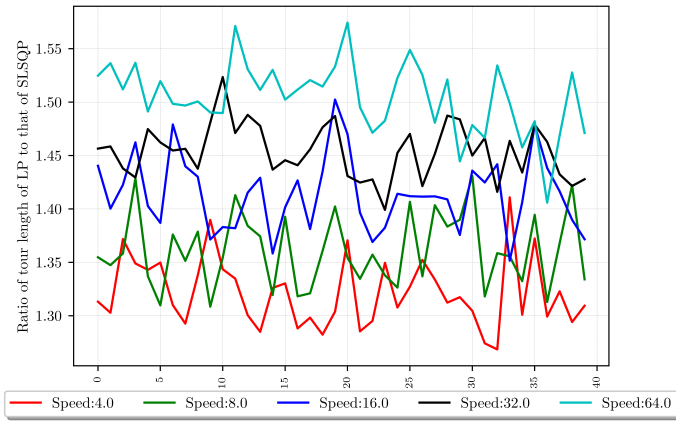


Fig. 3: Comparing lengths of tours obtained from LP and SLSQP

IV. Two heuristics

A. Greedy Heuristic

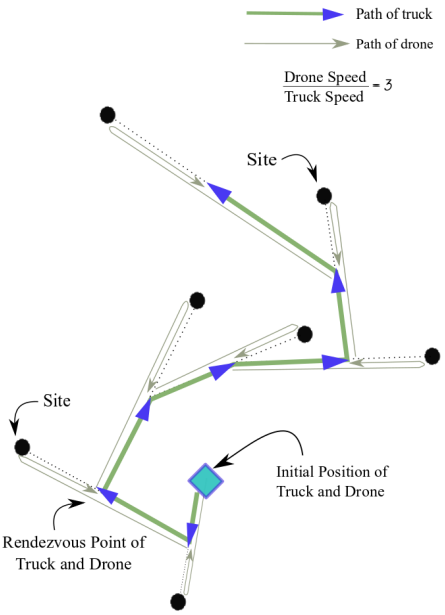


Fig. 4: The Collinear Horsefly problem

We first introduce a special case of the Horsefly problem, which we call *collinear-Horsefly*. Here, the objective function is again to minimize the tour-length of the drone with the additional restriction that the truck must always be moving in a straight line towards the site on the line-segment joining itself and the site, while the drone is also restricted to travelling along the same line segment. See Figure 4 for an example instance of this problem, for $\varphi = 3$. With this additional restriction, the possible rendezvous points for the truck and the drone becomes finite. We show that an optimal (unrestricted) Horsefly solution can be converted to a collinear-Horsefly solution, at a constant factor increase in the makespan.

Similar to the nearest-neighbor insertion heuristic for the standard TSP, a natural greedy strategy can be formulated for the collinear-Horsefly problem; the truck always moves towards an unvisited site nearest to its current position, while the drone takes off from the truck, services the site, and flies back towards the truck along the line joining the truck and the site again as shown in 4.

Once the ordering of the sites is determined by the heuristic, we use the fixed-ordering optimization program (using a convex program in the L_2 case, or a linear program in the L_1 case) to compute the best route for the truck (and thus for the drone), using the given ordering.

B. Clustering-Based Heuristic

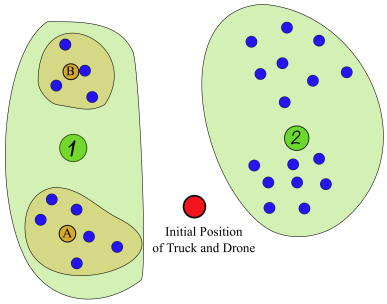


Fig. 5: The first two steps in the k2means heuristic for the Horsefly problem.

We describe another heuristic, which we call the "k2means" heuristic: Given the set S of n sites, we first compute the 2-centers along with associated cluster points for each of the 2-centers. We then use the exact algorithm to decide which center (and hence cluster) to visit first.

In the above figure, for instance, we decide that the truck and drone will coordinate to visit all of the sites in the left cluster first and then the right. Within the chosen cluster, we again compute the 2-center and then use the exact algorithm for 2 points to decide which sub-cluster should the truck and drone visit first. For instance, in the example above, the heuristic decides to visit all the sites associated with the 2-center A and then the 2-center B.

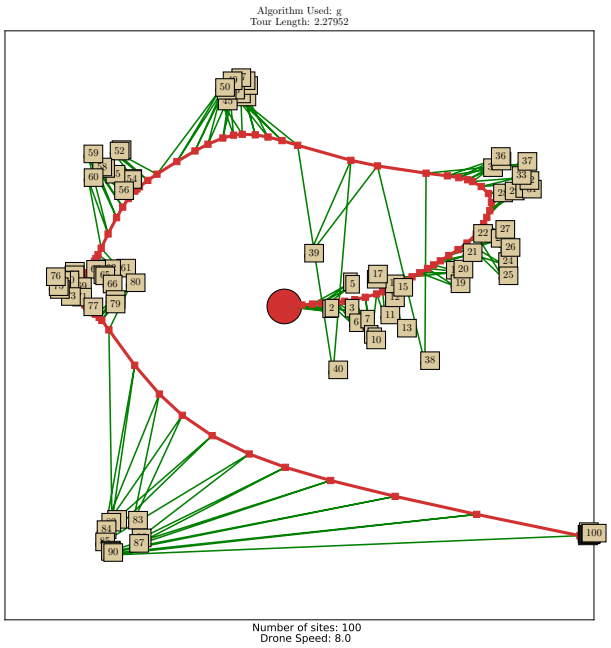
We keep doing this recursively for each cluster, until we reach a cluster size of 1 or 2.

In the above figure, once we finish visiting all of the sites in the left cluster, we use the ending point of this subtour as the initial point of the truck and drone to decide the order in which we must visit the sites in the right cluster.

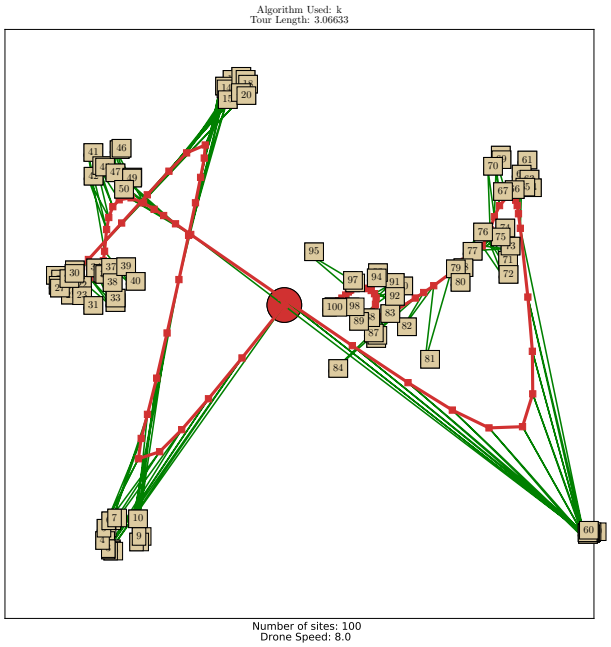
Once we finish calculating the order of the sites according to the heuristic, we discard the computed path, and use the non-linear solver to compute the optimal truck (and drone) path for this ordering.

C. An Example

Figure 6 compares two heuristics on one example of 100 sites distributed inside the unit-square, with $\varphi = 8$.



(a) Tour obtained with the greedy heuristic.



(b) Tour obtained with the k2means heuristic.

Fig. 6: An example of running the greedy and k2means heuristic on 100 sites

D. Comparing tour-lengths for the two heurisitcs

From Figure 7, we notice that the greedy heuristic consistently outperforms the k2means heuristic by a slowly growing function of n. The same behaviour was observed for other speed-ratios.

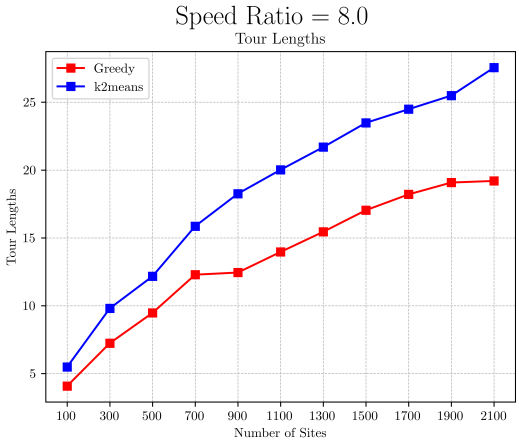


Fig. 7: Comparing the greedy and k2means heuristic

REFERENCES

[1] *https://www.businessinsider.com/amazon-and-ups-are-betting-big-on-drone-delivery-2018-3*

[2] UPS tests residential drone delivery. *https://youtu.be/P5hQHBNpd7s*

[3] John Gunnar Carlsson, Siyuan Song *Coordinated Logistics with a Truck and a Drone* Management Science, 2017, *https://doi.org/10.1287/mnsc.2017.2824*

[4] Ha, Q. M. and Deville, Y. and Pham, Q. D. and Hà, M. H. *On the Min-cost Traveling Salesman Problem with Drone*, *arXiv preprint arXiv:1512.01503*, 2015

[5] Campbell, James F and Sweeney, DC and II, Zhang J, *Strategic design for delivery with trucks and drones* Technial Report, 2017

[6] Chao, I.M. *A tabu search method for the truck and trailer routing problem* Computers & Operations Research, Elsevier 2002 volume 29, number 1, pages 33-51

[7] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html

[8] https://www.mosek.com/

[9] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html