Computational Geometric Approaches to Geospatial Optimization Problems

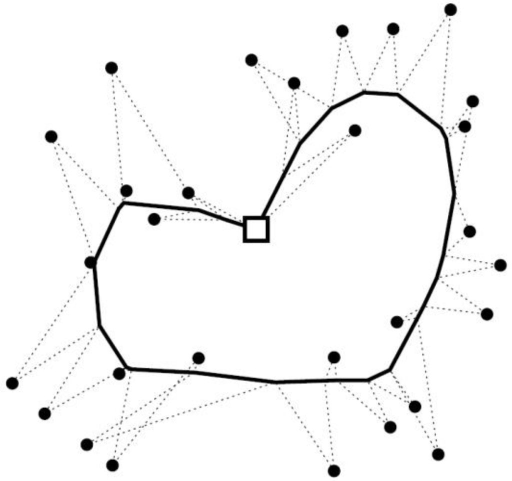
John Gunnar Carlsson, University of Southern California
Joseph S. B. Mitchell, Stony Brook University

Update: September, 2018

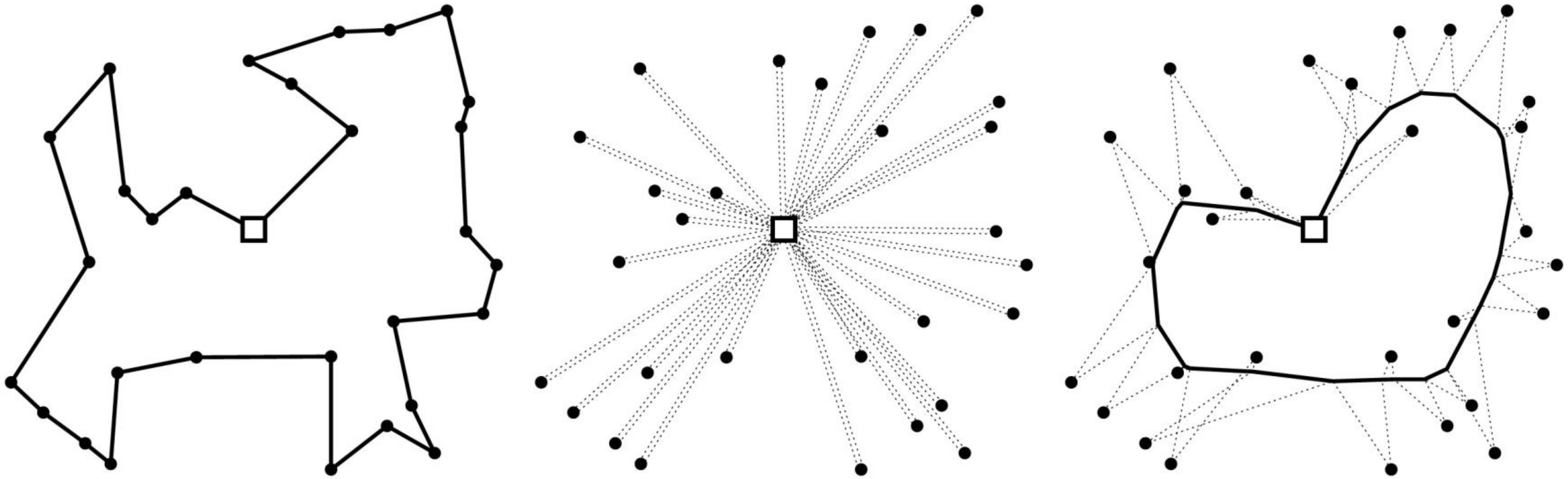
Challenges of Target Problems

- High-dimensional: many degrees of freedom, decision variables; infinite-dimensional problems
- Dynamic: trajectory optimizations in space-time, time-varying data
- Stochastic: uncertain data
- Combinatorially hard (NP-hard)
- Multi-scale: data at many different scales
- Non-convex
- Online, real time decisions may be required

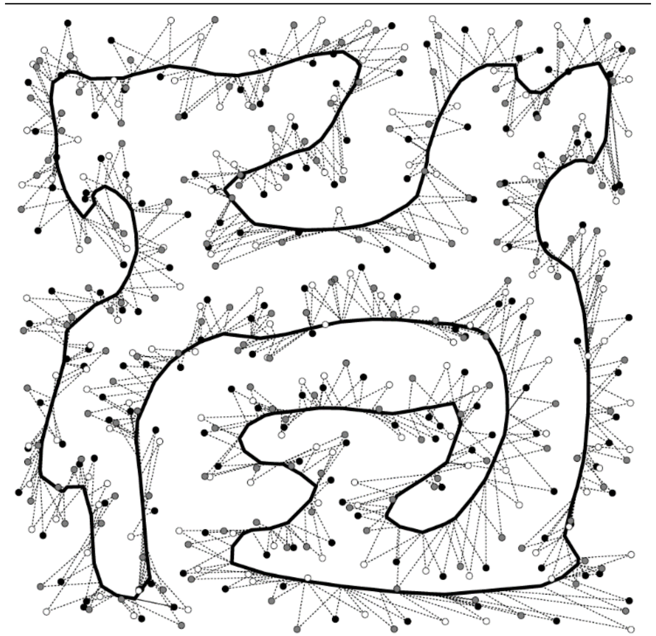
Multi-level routing – “HorseFly” problems



- Drone picks up a payload from a truck, which continues on its route, and after a successful delivery, the drone returns to the truck to pick up the next payload
- Truck is an “aircraft carrier”; does not stop at targets
- Computing the most efficient route is challenging because we have to coordinate both vehicles simultaneously
- For a fixed target sequence, the problem is a ***Second Order Cone Program (SOCP)***
- For a fixed truck route, the problem is a geometric scheduling problem



(a) Truck only (b) UAV only (c) Truck + UAV



500 customers, 1 truck, 3 UAVs

[Carlsson, Song]

Truck+Drone ("horsefly") Problem: Update

- Goal: Provable approximation methods that scale, work well in practice, and give new insights into coordinated vehicle optimization problems.

An asymptotic routing scheme

- Suppose that all sites P_i are independent and identically sampled from a smooth probability distribution f in a compact planar region
- We have found a method for routing the truck and UAV that performs within a constant factor ~5 of optimality, with probability one, ***in the asymptotic limit*** as $n \to \infty$
- Key idea is: use simple lower bounds to relate the horsefly tour to related problems (k-medians, TSP); then demonstrate that it is unlikely that the related problems' cost happens to be less than $c\sqrt{n}$, for suitable c
- E.g., "if there's a really good horsefly tour, then there's also either a really good TSP tour or a really good k-medians solution"

An asymptotic routing scheme

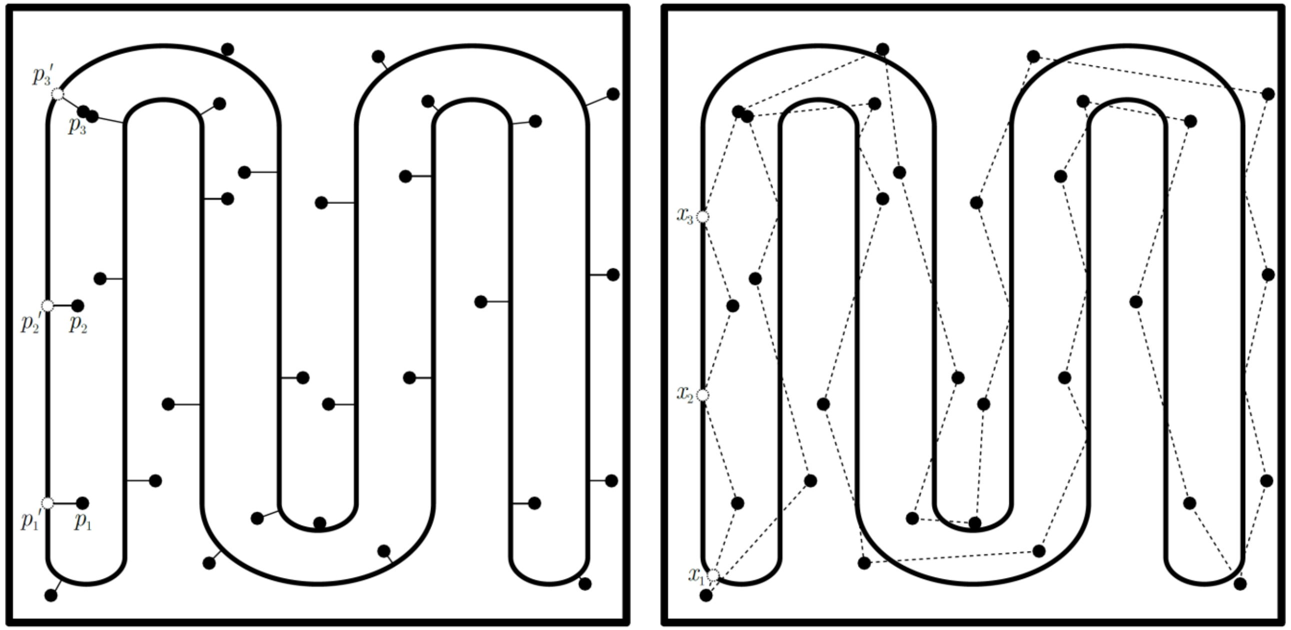
- The analysis extends when we have multiple drones: the key quantity of interest is

$$\sqrt{\frac{Truck\ speed}{(Drone\ speed) \times (\#\ drones)}}$$

- As a side consequence, we have obtained upper and lower bounds for the ***asymptotic Euclidean k-medians constant*** and improved the best lower bound of the ***asymptotic Euclidean sparse subset TSP constant*** from 0.05 to 0.2

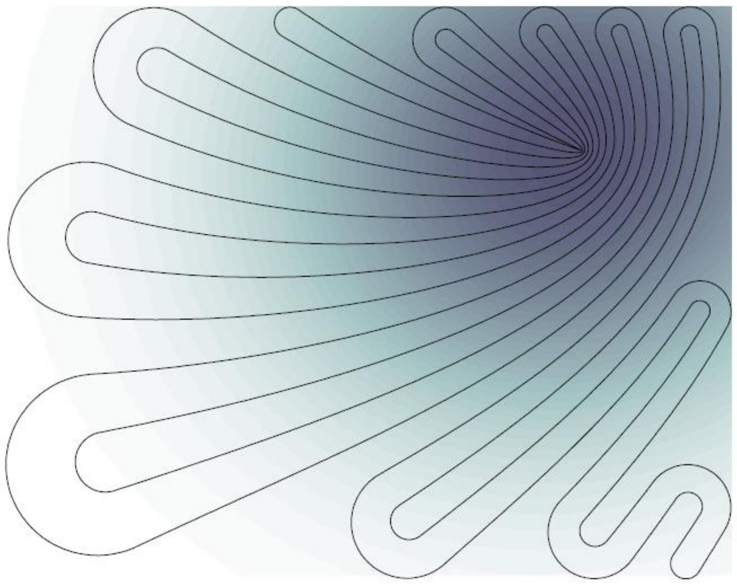
Routing scheme: “Heron’s heuristic”

- Pretend that we know the distribution is uniform, and zig-zag in a natural way:



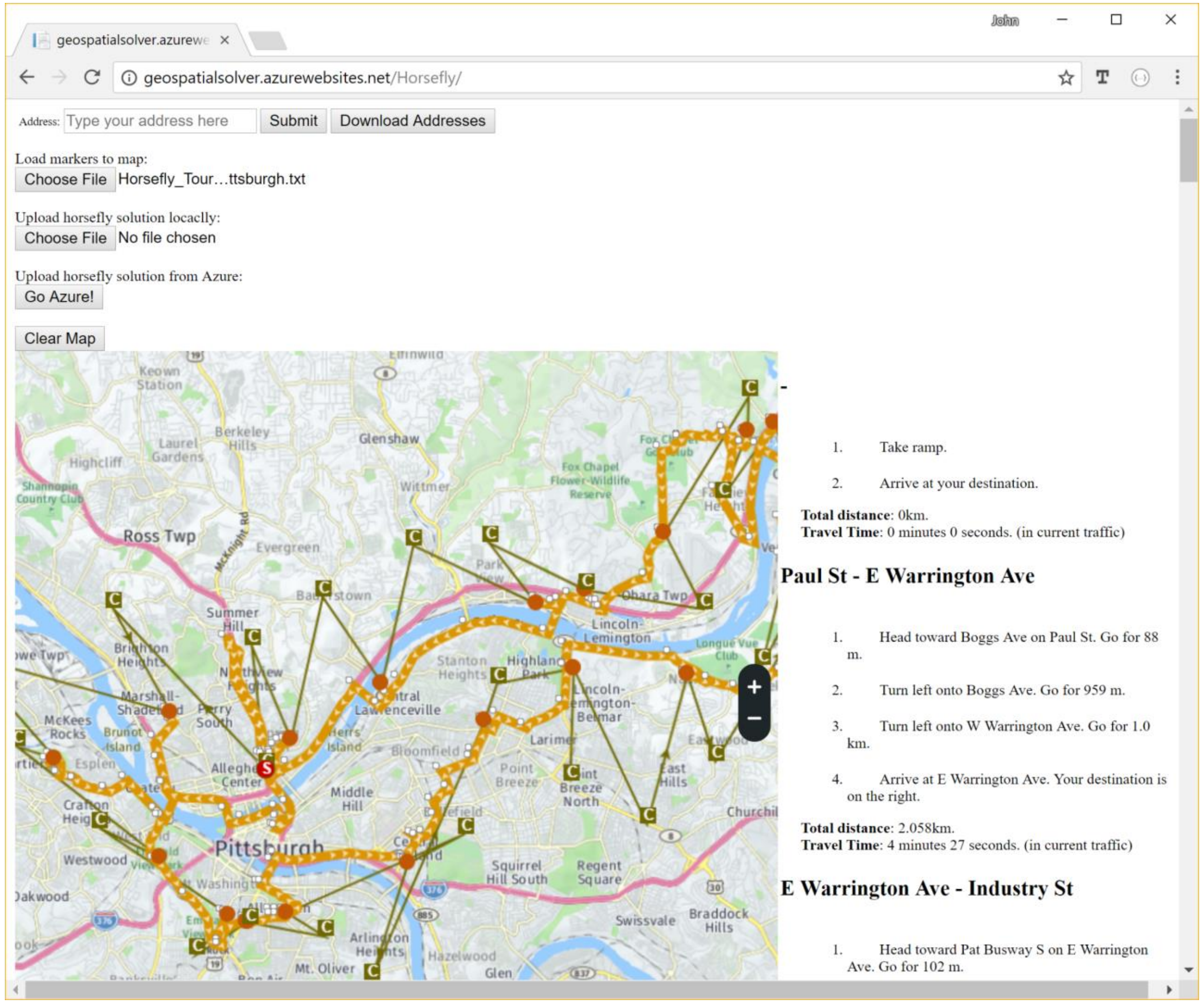
Routing scheme: “Heron’s heuristic”

- Pretend that we know the distribution (but it’s not uniform), and zig-zag in a natural way:



- Finally, use the fact that we can estimate the distribution arbitrarily well as $n \to \infty$

Cloud service on Microsoft Azure



Truck+Drone ("horsefly") Problem Progress:

- Local optimality and properties of optimal solutions
- Approximation algorithms, with guarantees per instance (compare to asymptotic opt)
  - Polylog approx: first, based on one-of-a-set TSP
  - Log-approx: improvement, based on structure theorem, DP, binary space partitions
  - O(1)? Possible progress towards goal of constant-factor
  - PTAS?? Open, but not out of the question
- Variants of the problem:
  - Fixed truck route
  - Customers/targets have their own drones, sent to the truck along its route
- Experimental investigations

Stony Brook University

Truck+Drone ("horsefly") Problem:

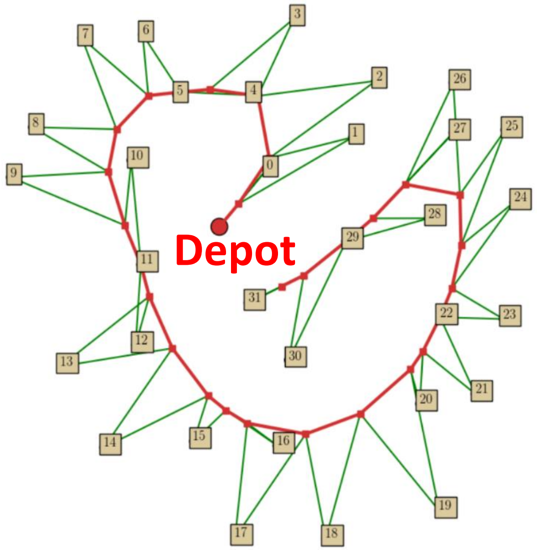


Figure 1: Truck (Red) and Drone(Green) servicing 32 sites with speed of drone $\varphi = 4$

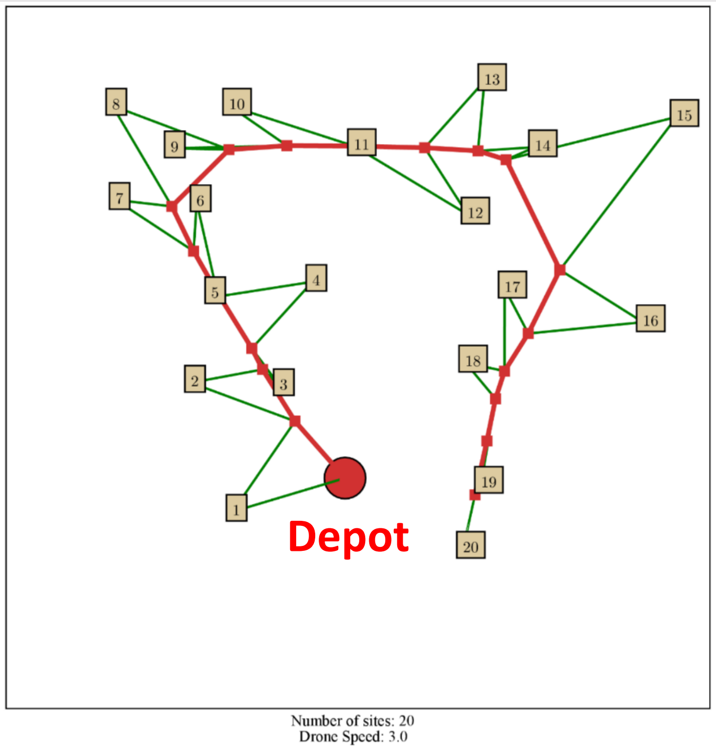
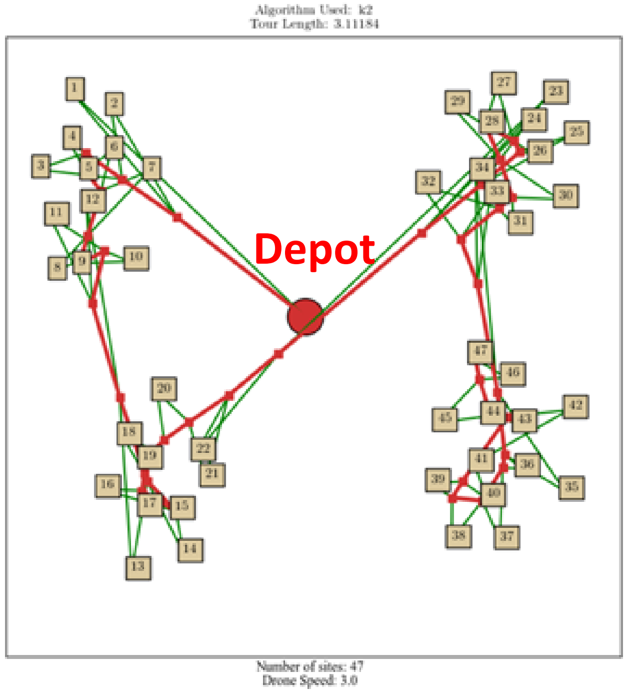


Figure 3: Using the TSP ordering to extract a routing for the truck and drone



: Running the k2 means algorithm algorithm for 47 sites with $\varphi = 3$

Case: Completion when drone returns to truck after last delivery. (implies truck/drone never wait)

Alternative Case: Completion when truck/drone return to depot, after all deliveries. (truck might pause/wait for drone)

[images: SBU student Gaurish Telang testbed]

Stony Brook University

Local Optimality, First Observations

- Truck and drone routes are polygonal (Euclidean metric)
- Given fixed target sequence, the problem of computing an optimal truck route to min makespan is a Second-Order Cone Program (SOCP)
- An optimal solution can have drone route crossing truck route, and drone route crossing itself

Unsure: Might the opt truck route self-intersect? We have not seen it, but exchange argument not yet working to prove it

- Simple/trivial ($r+\varepsilon$)-approximation (r=speed ratio = v_{drone}/v_{truck}):
  - Just have truck+drone travel to each target together
  - Use PTAS ($1+\varepsilon$)-approx for TSP of targets

Thus, goal is to get *below* factor r, ideally a factor O(1) indep of r .

- For approximation purposes, to get O(1)-approx in Euclidean problem, it suffices to solve/approximate the L_1 metric version (give up factor sqrt(2))

  Fixed target sequence problem is LP (instead of SOCP)

Math Program for Given Ordering:

$$\min_{(x_1,x_2,\ldots,x_{2n})\in\mathbb{R}^{2n}} \sum_{i=1}^{n} ||X_i - X_{i-1}||$$

subject to the n constraints

$$||X_i - X_{i-1}|| = \frac{||X_{i-1} - P_i|| + ||P_i - X_i||}{\varphi} \qquad \text{for } \; i \in \{1, 2, \ldots n\}$$

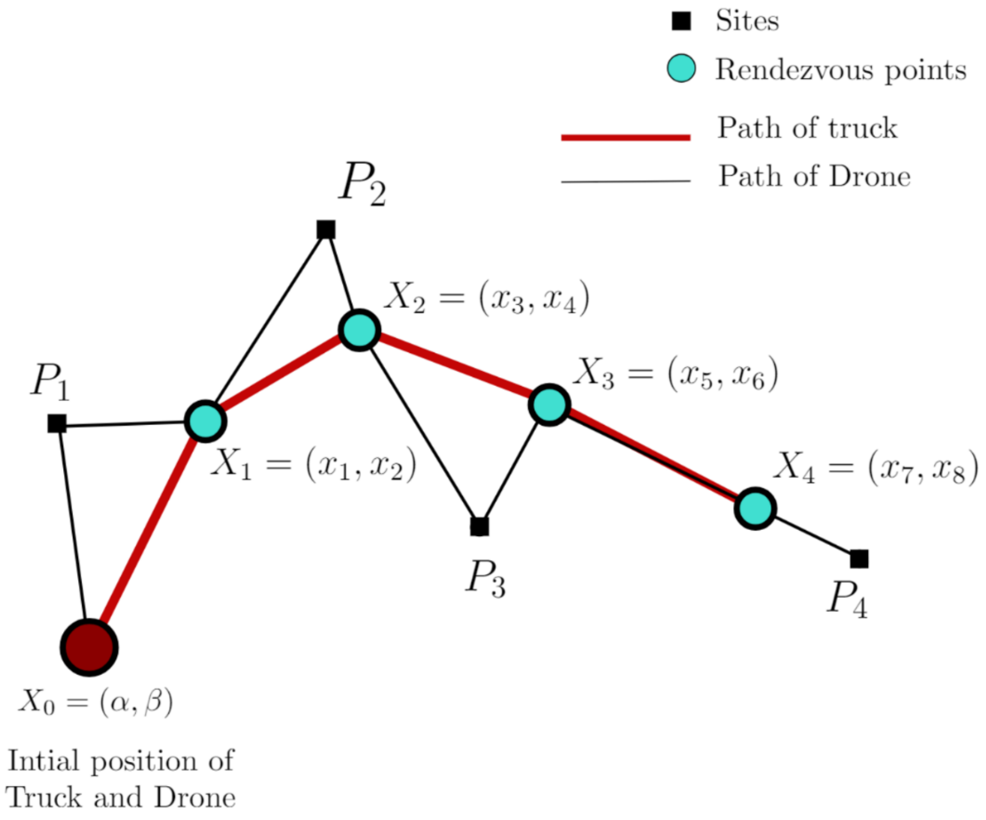


Figure 2: Solving the Horsefly Problem when the order of visitation is given

Speed ratio $r = \varphi$

For these experiments, used the Sequential Least Squares Programming (SLSQP) package in SciPy to solve.

NOTE: For this problem (given ordering), we can give an O(1)-approx, combinatorial algorithm

A Local Optimality Condition

(assuming no waiting/pausing)

Note that

$$\frac{a+b}{\varphi} = c$$

Also note by the sine rule

$$\frac{a}{\sin(\alpha)} = \frac{b}{\sin(\beta)} = \frac{c}{\sin(\gamma)}$$

which implies

$$\frac{a+b}{\sin(\alpha)+\sin(\beta)} = \frac{c}{\sin(\gamma)}$$

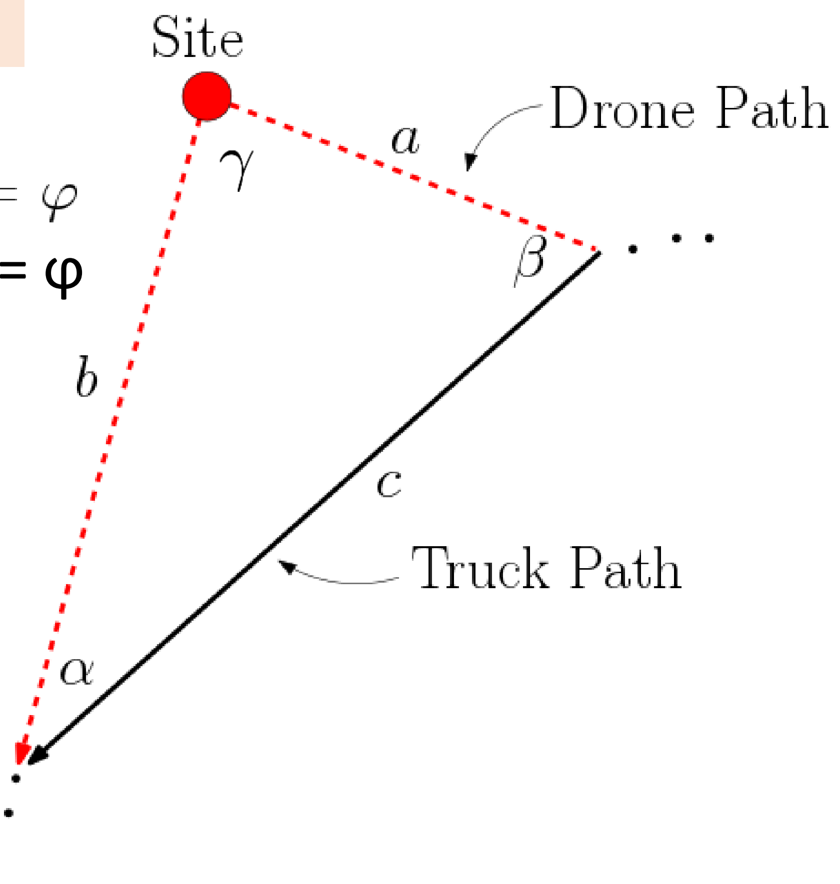
$$\frac{a+b}{\frac{\sin(\alpha)+\sin(\beta)}{\sin(\gamma)}} = c$$

But since $(a+b)/\varphi = c$ and $\sin(\theta) = \sin(\pi - \theta)\ \forall \theta \in \mathbb{R}$

we have

Drone Speed $= \varphi$

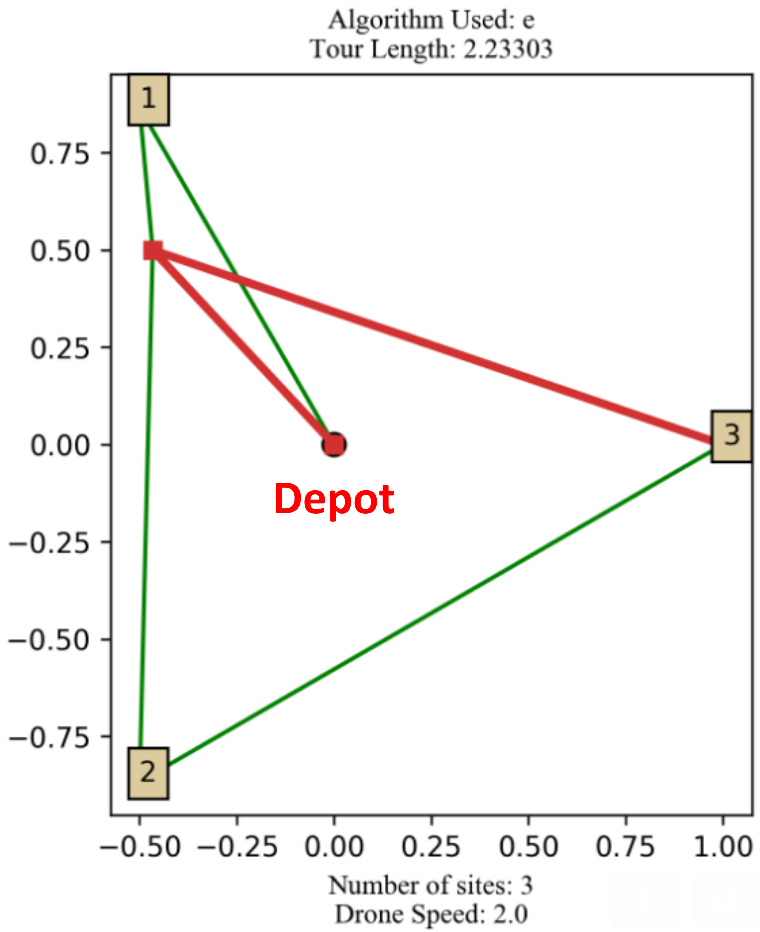
Speed ratio $r = \varphi$

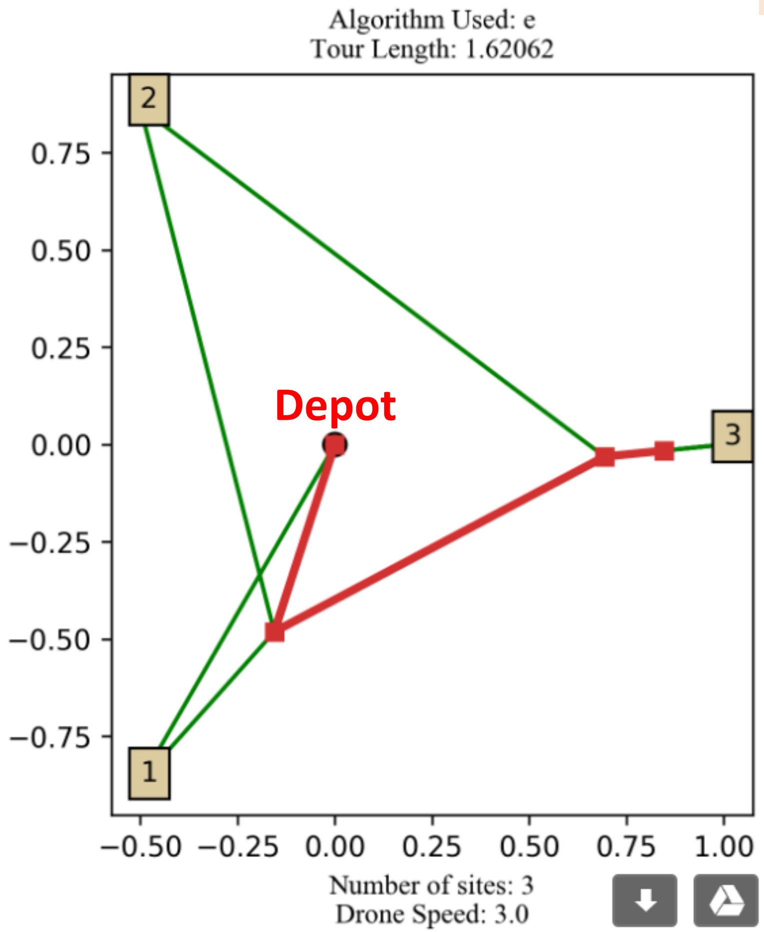


$$\varphi = \frac{\sin(\alpha)+\sin(\beta)}{\sin(\alpha+\beta)}$$

Example: In Opt, Drone Route May Self-Intersect, and Cross Truck Route

Case: Completion when drone returns to truck after last delivery





Approximation Algorithms

- A first provable approximation (polylog factor) algorithm for general instances:
- New: O(log n)-approximation
- Goal: Determine if O(1)-approx, or even PTAS, is possible

Natural candidate heuristic: greedy insertion

For a solution tour (truck, drone) on *i* targets, examine each of the remaining n-i targets and insert (cheapest local insertion) the one that lengthens the makespan the least.

Q: Is this a provable approximation? (for usual TSP, it is 2-approx)

Polylog Approximation

- Localization: Identify the approx bounding box, B, of an opt truck route
- Lay a n^2-by-n^2 size grid, G, on B, to get grid cells that are fine enough: Snapping Opt to have truck have vertices on grid preserves optimality (approx)
- For each target p_i, consider the set P_i of n^4 pairs (p_i,g), for each of n^4 grid points g for the truck-drone rendezvous point.
- Distance from (p_i,g) to (p_j,g') is

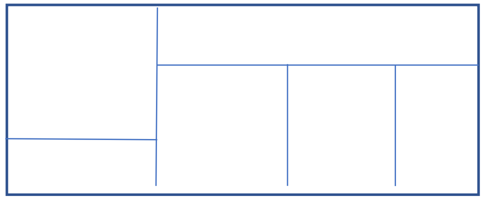
max{time for truck to go from g to g', time for drone to go from g to p_j to g'}

- Do one-of-a-set TSP on the sets P_i : polylog (roughly $\log^3$) approximation

Improvement: O(log n)-approx

- Dynamic Program: Subproblem is a rectangle R, around which the truck travels the full perimeter.

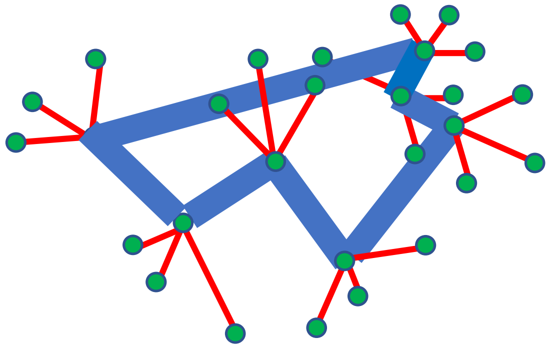
  Optimize: Find cheapest "BSP" truck network, with "spokes" to all target sites, weighting the length of the truck network by r, the speed ratio (and spokes with weight 1)

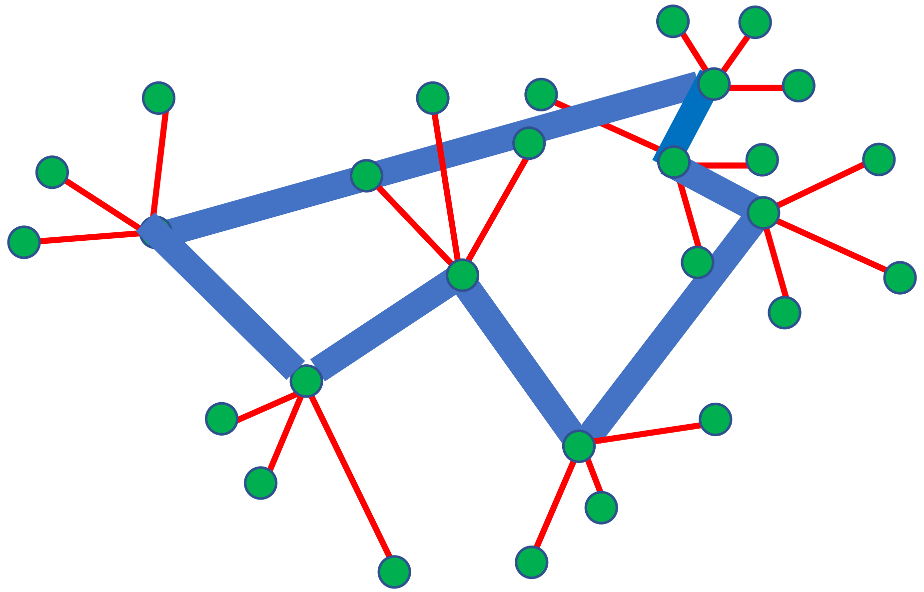


BSP

Proof of approx factor:

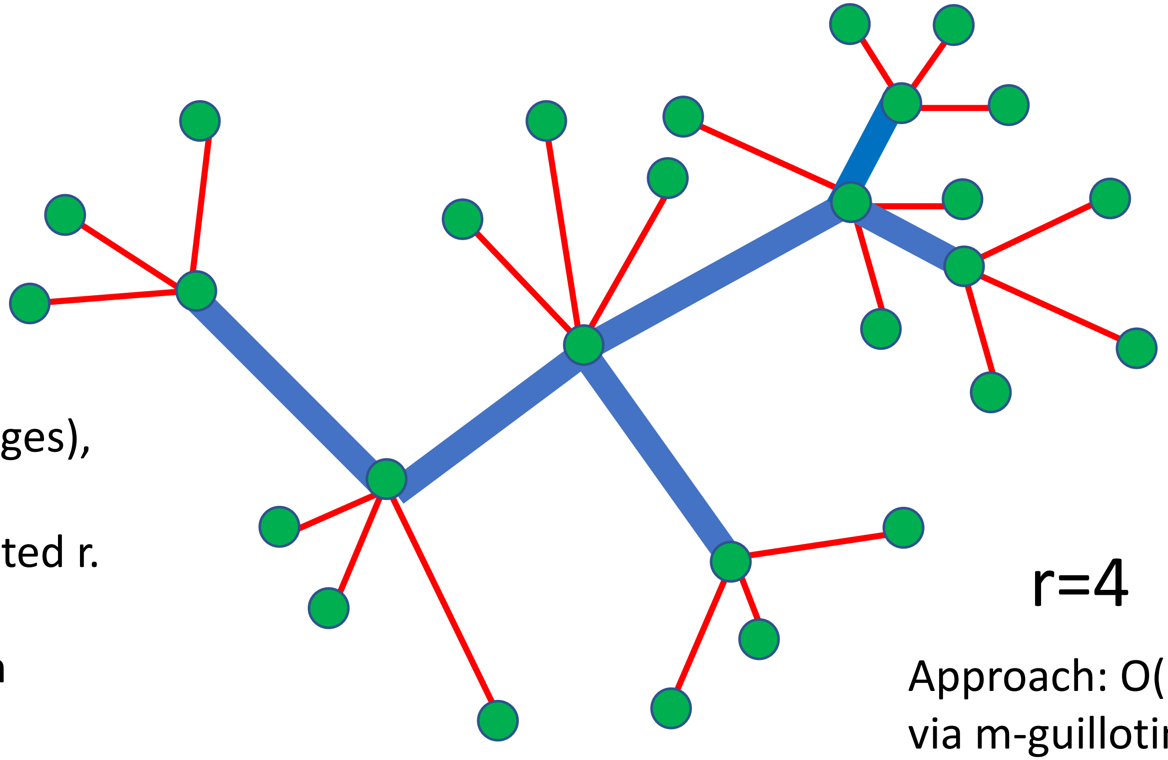
- Convert drone route to be a set of doubled "spokes" attached to truck route at pause points (truck waits) O(1) factor
- Convert Opt routes to be rectilinear (L_1) O(1) factor
- Augment truck route to be a BSP network (factor log dilation); note that resulting solution is among those searched by DP
- Solution recovery: From DP solution, at O(1) factor can retrieve a valid solution to original problem





Towards an O(1) Approximation

- Lemma: An α-approximation for the Weighted-Backbone-and-Spoke Spanning Tree (WBSST) problem implies an O(α)-approximation for Horsefly.



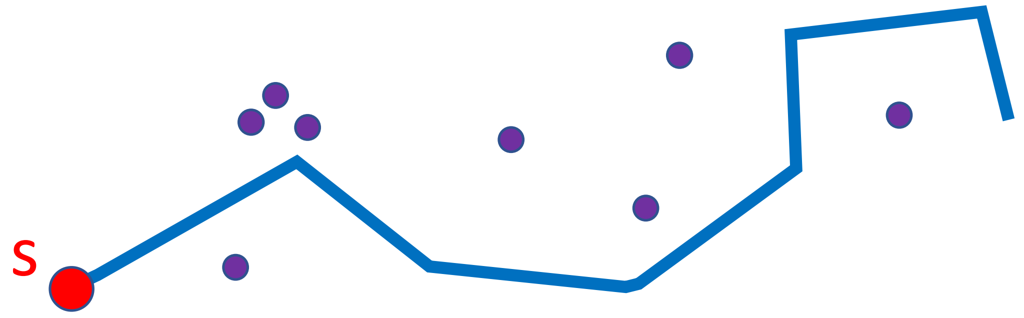
Cover sites with r-stars (weight 1 edges), linking the centers of r-stars via a spanning tree (blue) of edges weighted r.

WBSST: Compute a min-weight such spanning structure

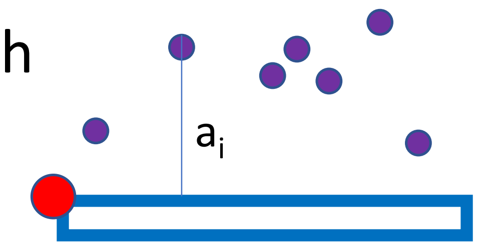
r=4

Approach: O(1)-approx for WBSST via m-guillotine methods

Fixed Truck Route Variant



Truck route is *given* and the goal is to schedule the speed of the truck (which may stop/start) and the drone routes to min makespan.



- Weakly NP-hard (from PARTITION), in general, for route=cycle
- Special case: Single segment path *st* (not cycle) for truck.

  Two subcases:

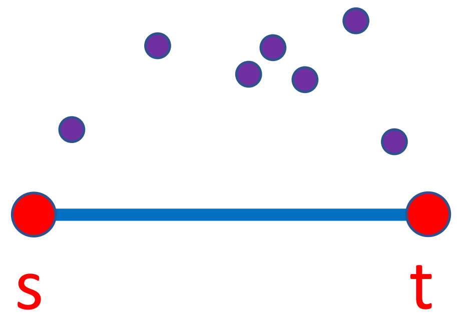
  (a) Completion when drone and truck both reach *t*

  - *(a) In L_1 metric, exactly solved: Fact (exchange argument): Targets visited in x-order*
  - *(b) In L_2 metric, complexity is open*

  (b) Completion when drone returns to truck after last delivery

  Issue: Even for L_1 metric, Opt may visit sites in order other than x-order

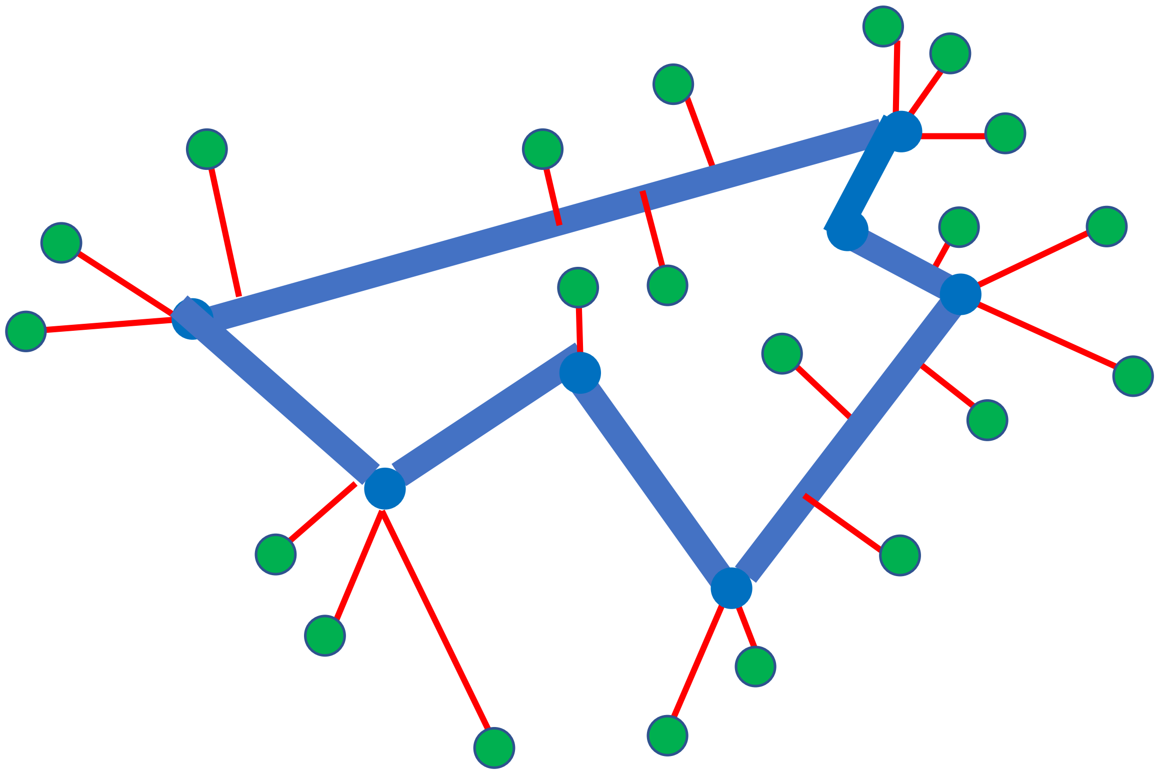
  Open: Complexity of problem, for either L_1 or L_2 metric



“Reverse Horsefly” Variant

Plan a route for the truck, but now there is a drone at each customer, which is sent to rendezvous with the truck to pick up the package. Goal is still to min makespan.

- In general, NP-hard (from TSP)
- Goal: Good approximations



Reverse Horsefly, Fixed Truck Route

The truck route is given, and each target customer has their own drone that can be sent to rendezvous with the truck, to get the package.

Easy "solution": Each drone simply flies to the nearest point on the truck route, and back. Easy to compute (e.g., Voronoi diagram induced by truck route)

Issue: This solution does not take into account that there is a minimum time window, *w*, necessary for a drone to dock, be loaded, and depart; the naïve solution may have thousands of drones try to arrive at once

Solution: Solve as a bipartite matching problem, matching target customers with time slots along the truck route, allowing multiplicity of slots (corresponds to the truck stopping). Bottleneck matching (for makespan optimization), or min-weight (min energy, total of all flights), or bicriteria version

Discretization of time slots – approx to continuous case

Some Experiments

- PhD Student Gaurish Telang (SBU)
- Exact solver (brute force – enumerate all target orderings), for testing purposes
- Utilize SLSQP solver for exact L_2 (convex, SOCP) optimization, for given ordering
- Utilize LP solver for L_1 optimization of routes, for given ordering
- Heuristics for determining an ordering:
  1. Greedy: increment the truck/drone routes to “closest” unvisited target
  2. K2-means clustering based heuristic
  3. TSP on target sites

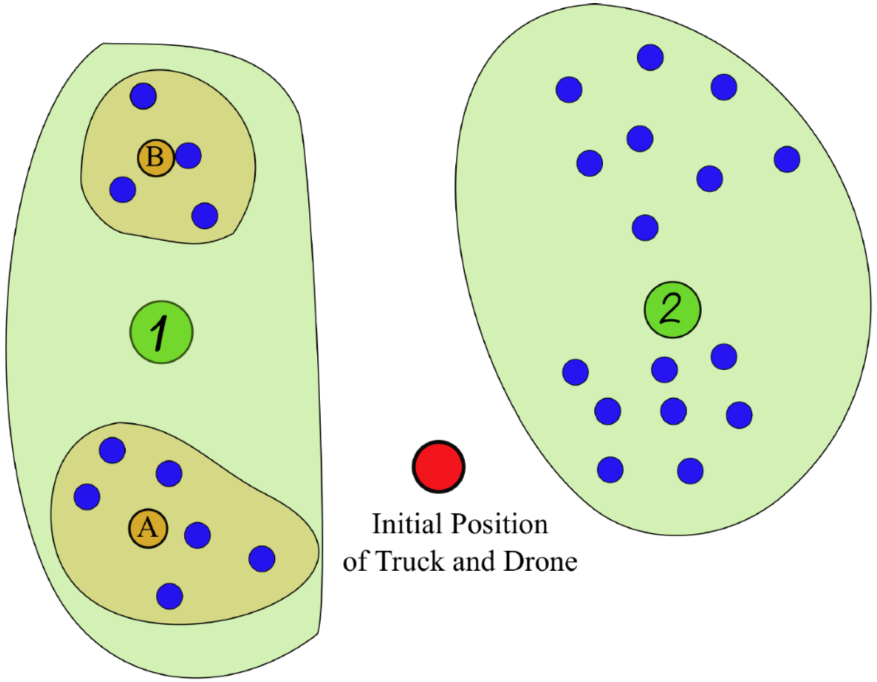
K2-means Heuristic to Obtain Ordering

In this algorithm, we use a k means black-box routine (from the scikit-learn Python package used extensively in machine learning) that computes exact or approximate k-centers. In these experiments, we set $k = 2$. Given a set of sites, we first compute the 2-centers along with associated cluster points for each of the 2-centers. We then use the exact algorithm described in a previous subsection to decide which cluster to visit first.

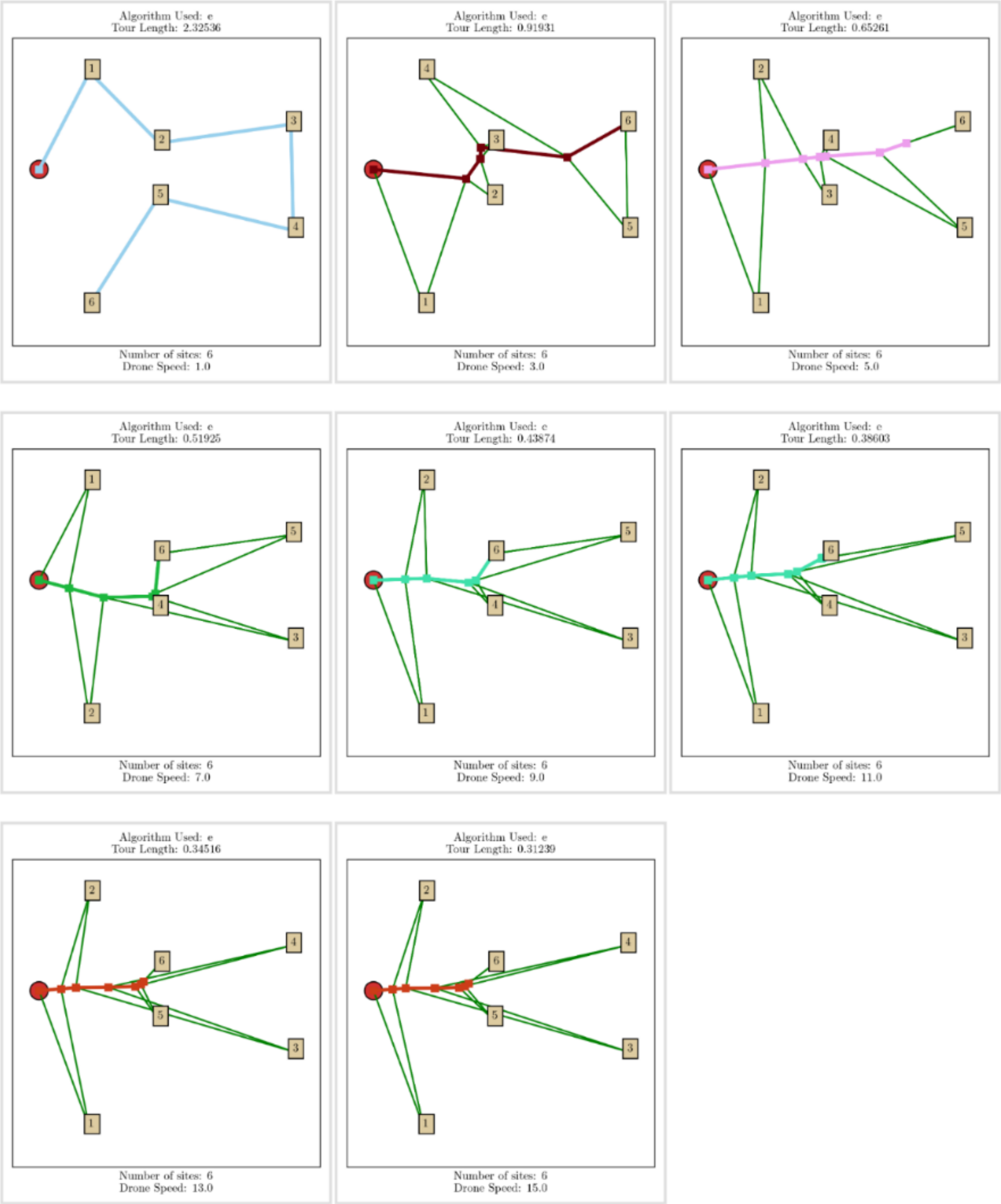
In the above figure for instance, we decide that the truck and drone will coordinate to visit all the sites in the left cluster first and then the right. Within the chosen cluster, we again compute the 2-center and then use the exact algorithm for 2 points to decide which sub-cluster should the truck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

We keep doing this recursively for each cluster, till we reach a case where the size of the cluster is 1 or 2.

In the above figure once we finish visiting all 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.



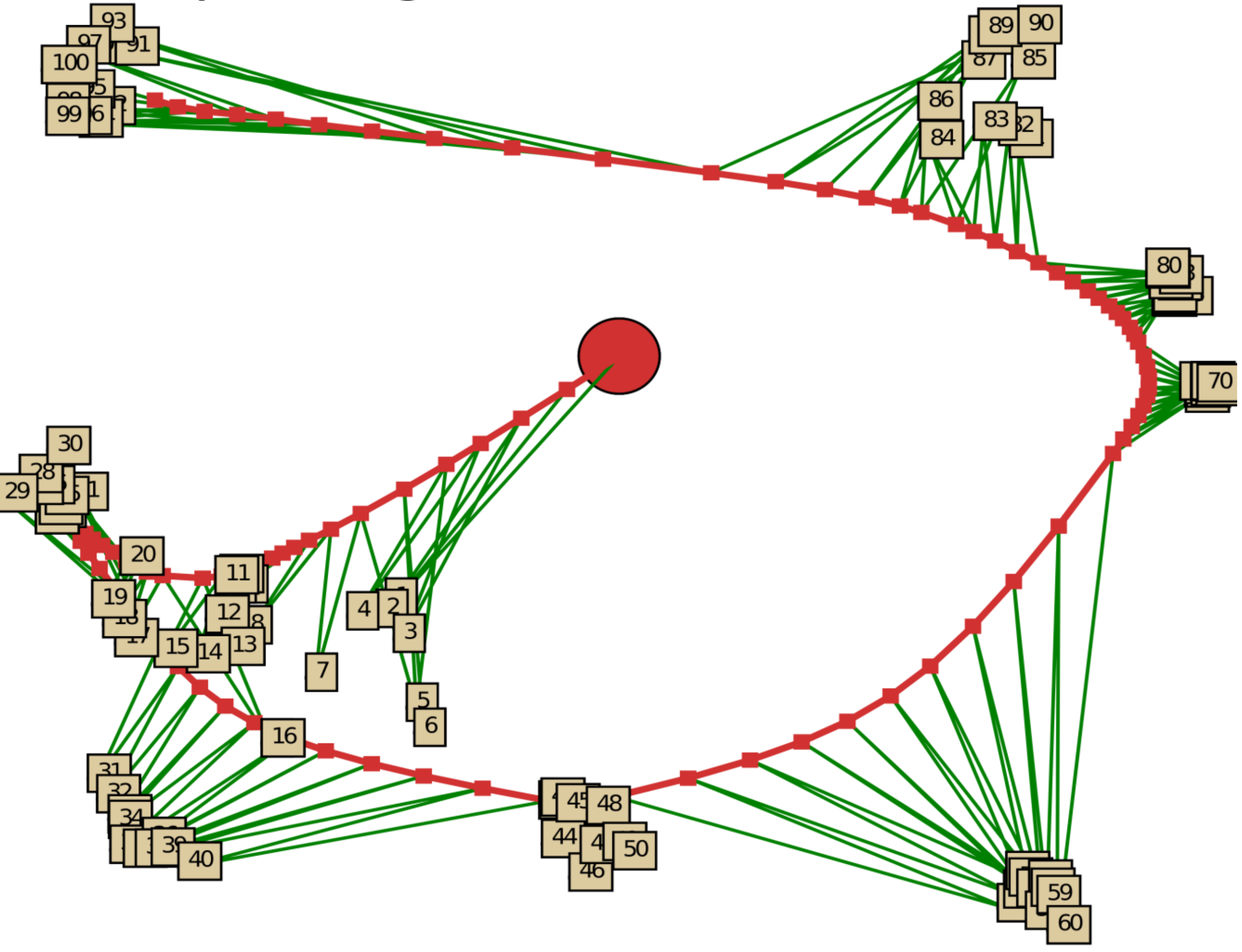
Example: Changes in (exact) OPT Truck Route as Speed Ratio Varies



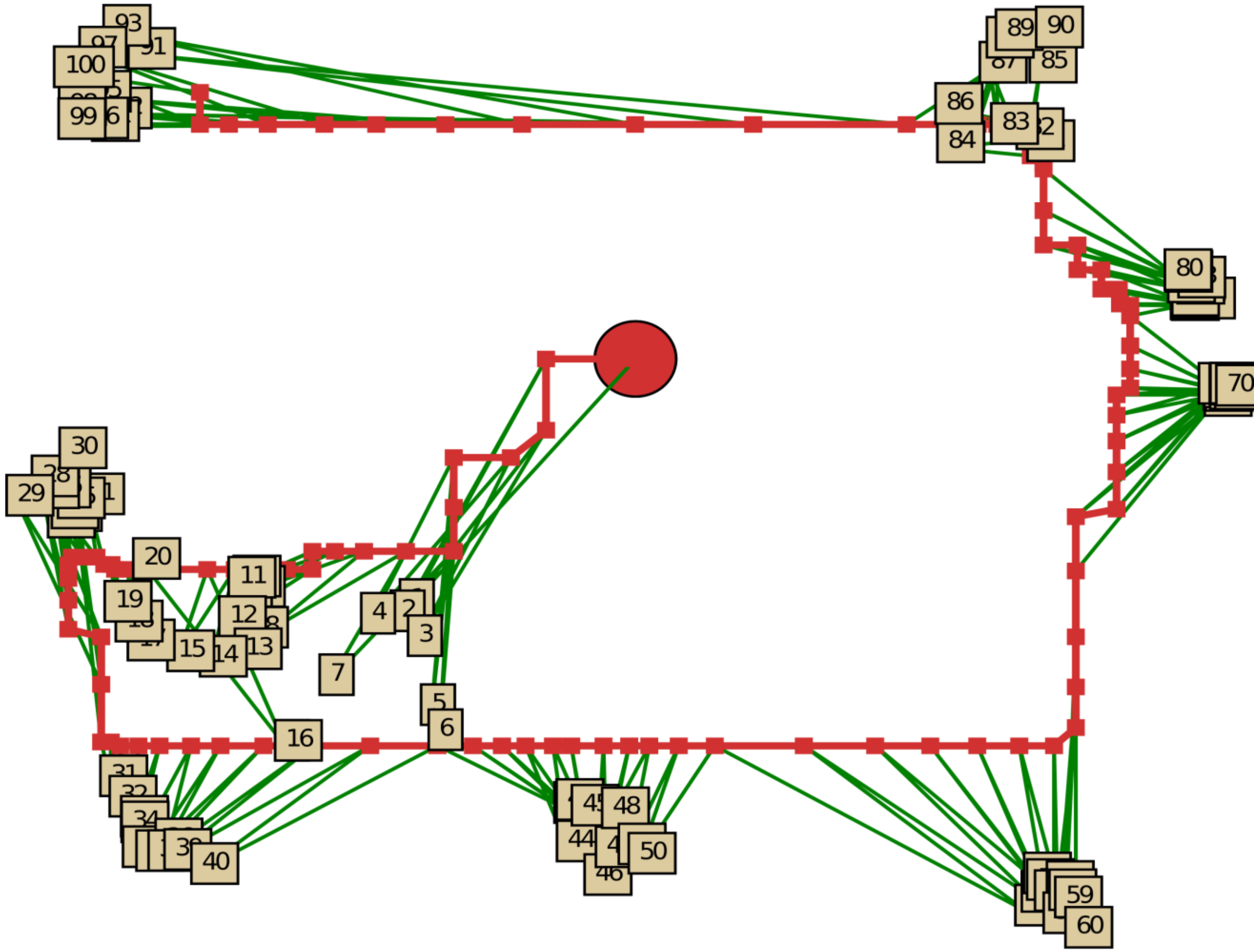
The full animation sequence from which the above frames have been sampled is available on GitHub at https://bit.ly/2uDpfyr

When an ordering of sites is specified:

Comparing Exact and an LP based solution for 100 sites $(\varphi = 10.0)$

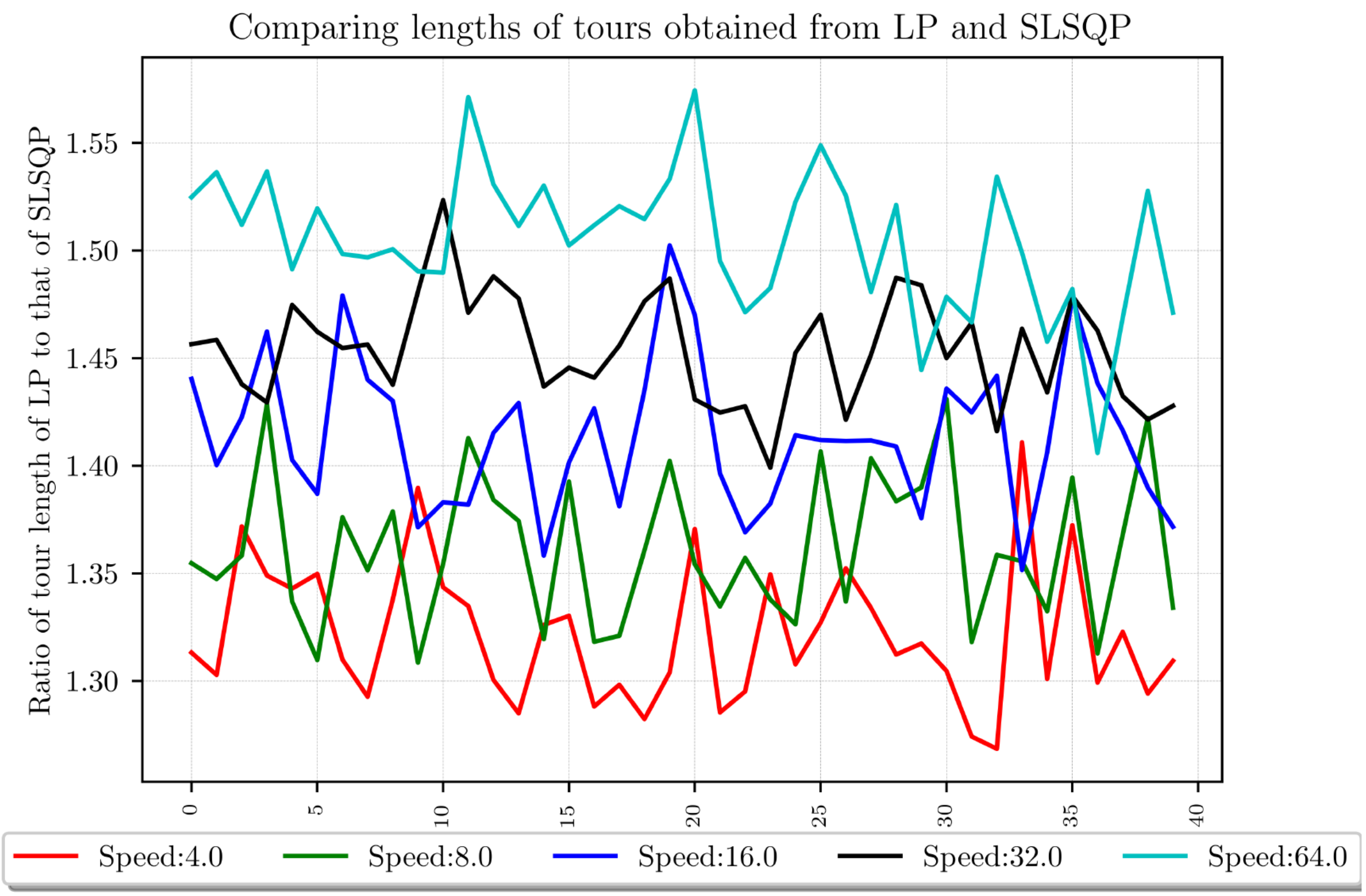


Greedy Ordering and SLSQP (Runs slowly)
Tour Length: 2.23



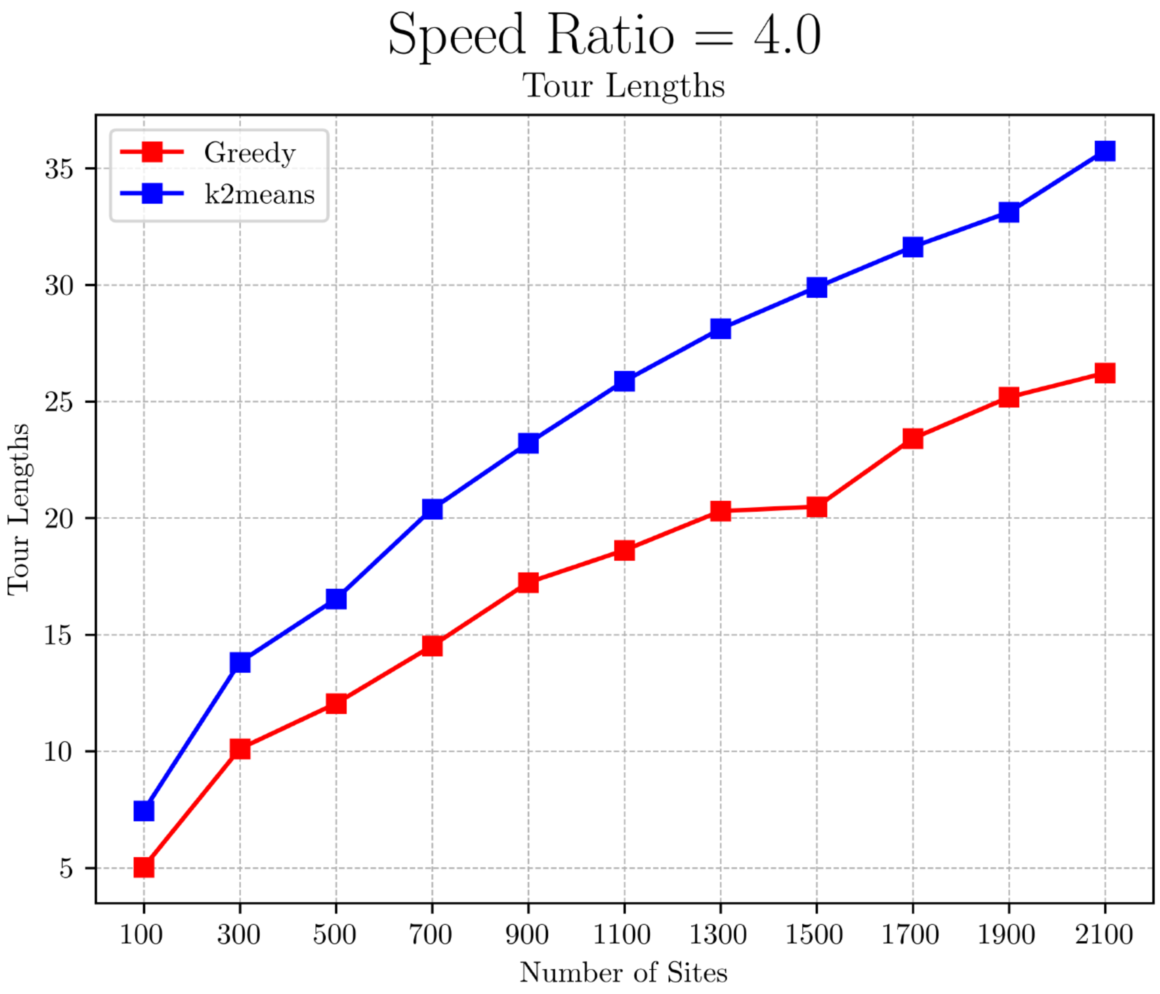
Greedy Ordering and LP (Runs quickly)
Tour Length: 3.08

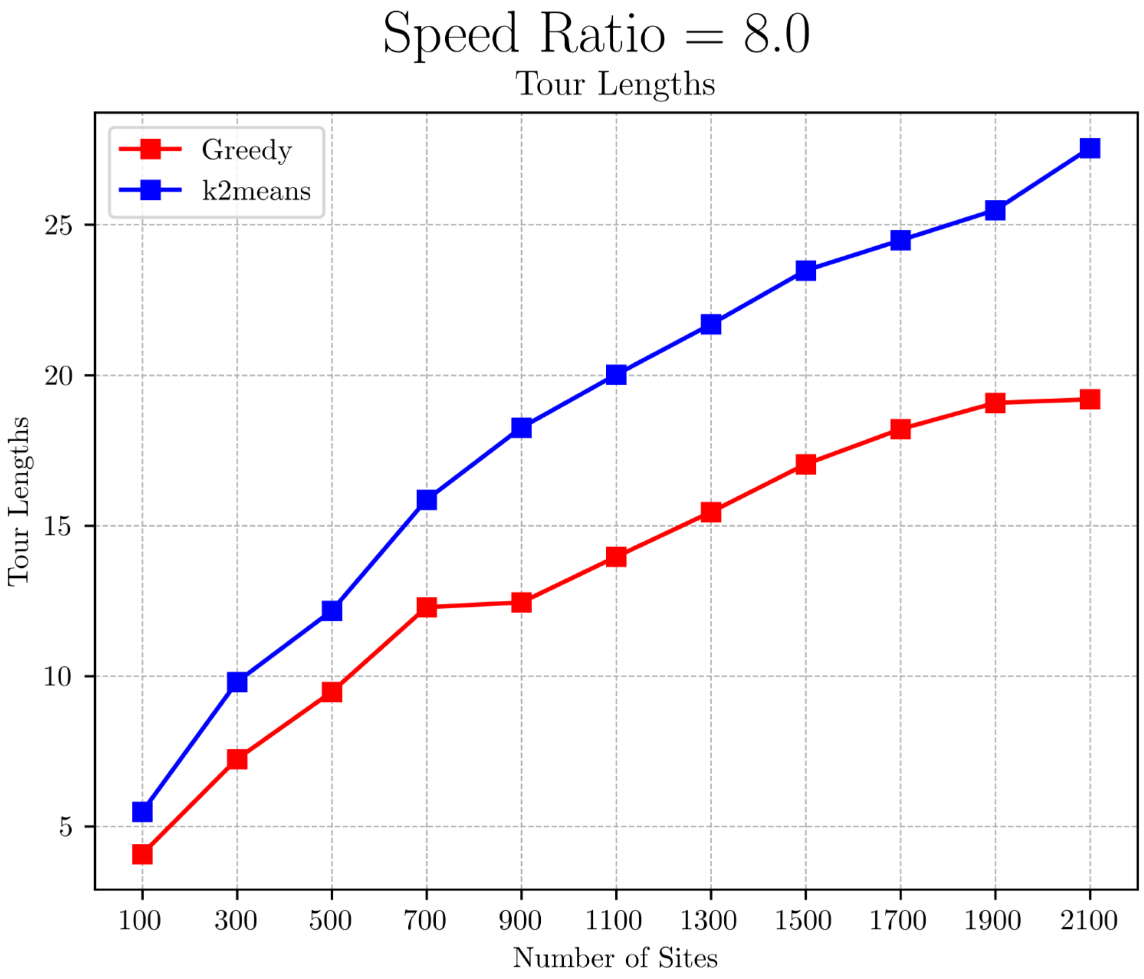
Ratios of lengths of tours: LP/Exact for given ordering



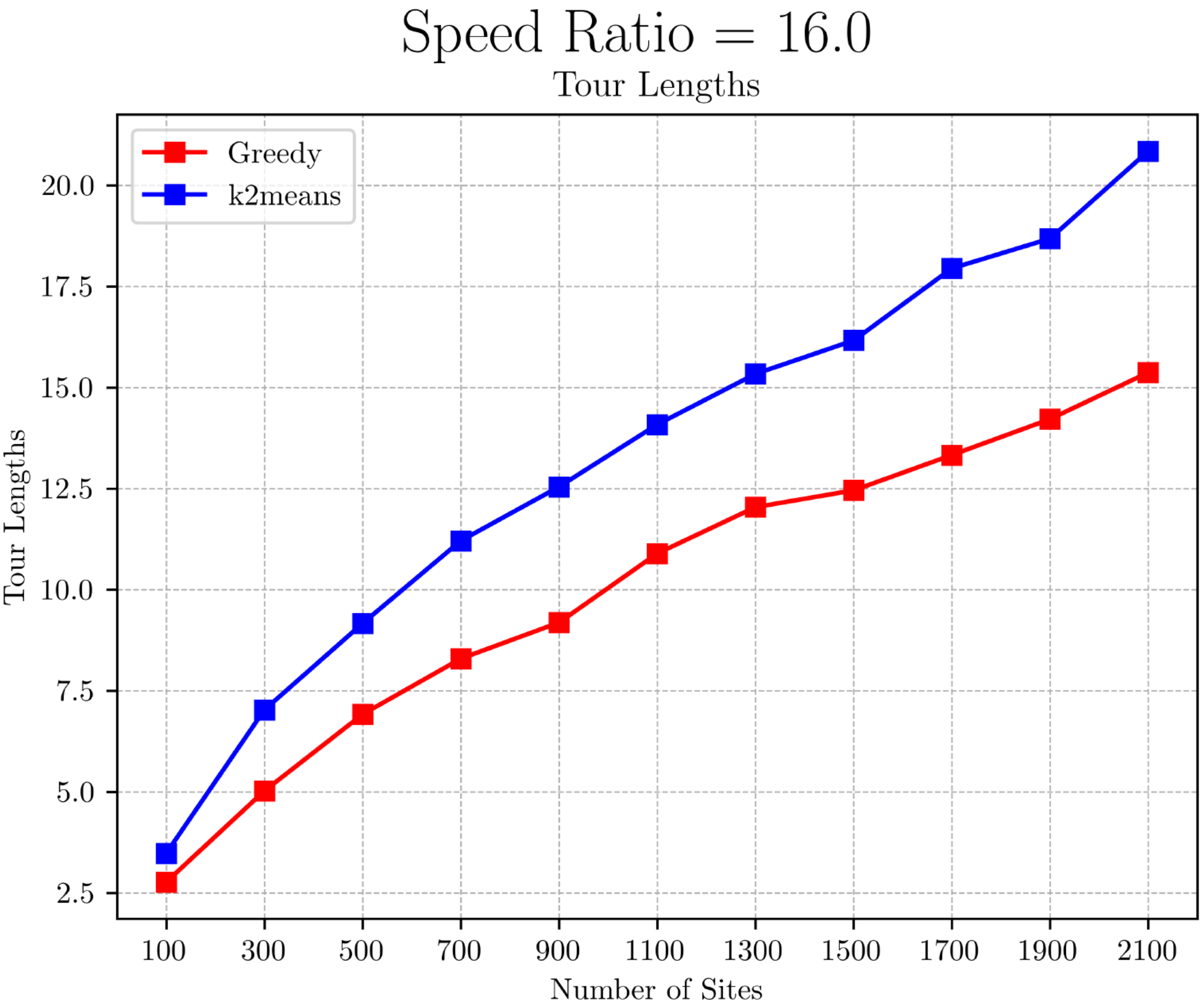
- 100 randomly chosen points in $[0,1]^2$
- 40 different runs of the experiment
- Greedy Ordering used
- Worst case ratio of the lengths seems to grow slowly with increasing drone speed φ

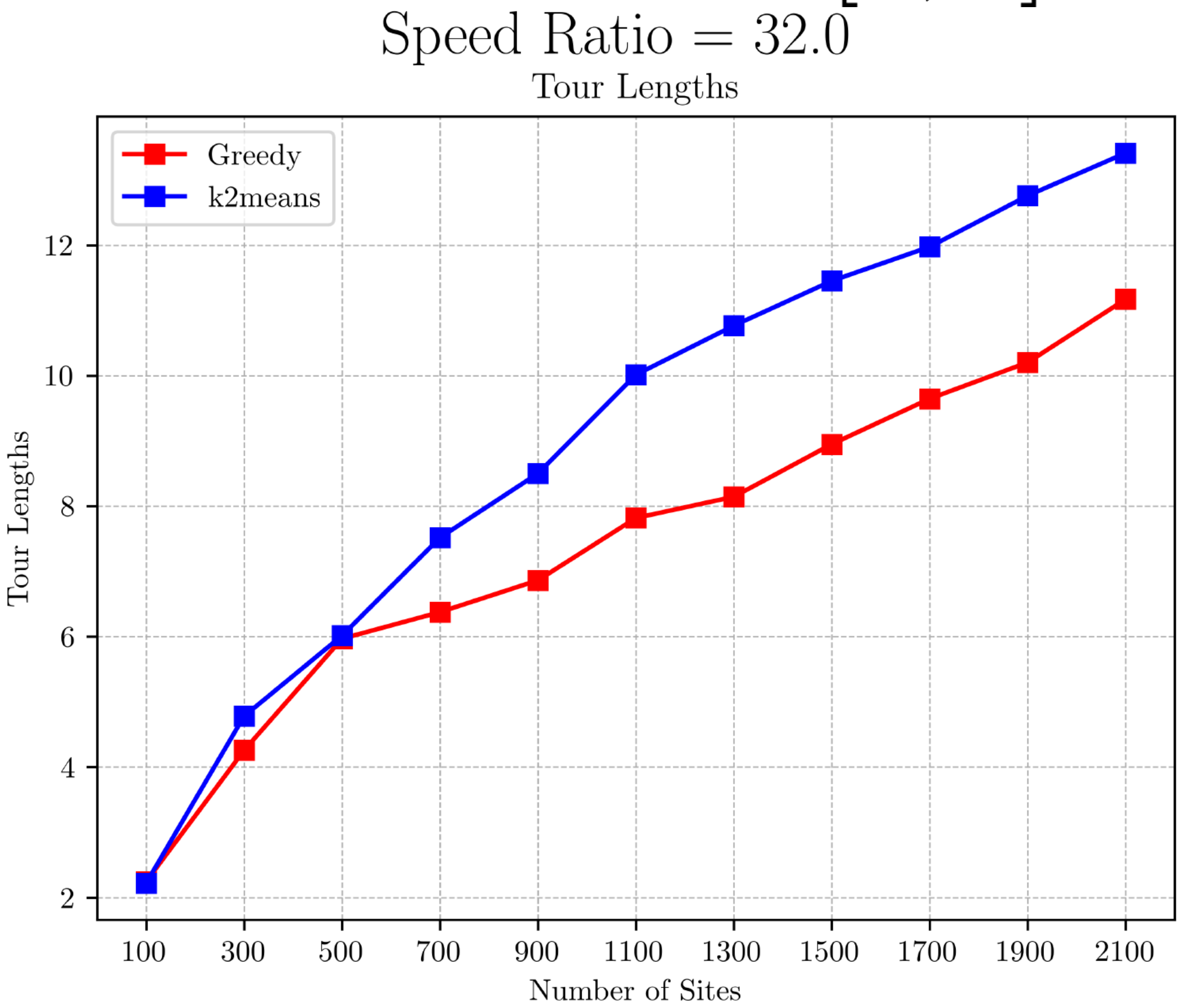
Comparing K2means and Greedy for a large number of sites uniformly distributed in $[0,1]^2$





Comparing K2means and Greedy for a large number of sites uniformly distributed in $[0,1]^2$





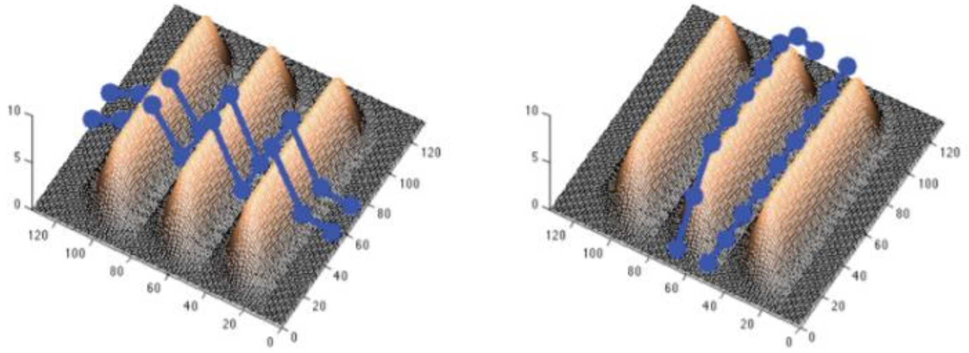
Next Steps

- Continue to resolve the approximability questions
- Generalize to k drones
- Add flight distance constraint (on drones)
- Other metrics (e.g., induced by road networks, instead of L_1, L_2)
- Online, distributed variants
- More experiments!

(newly underway, fall 2018)

Multi-vehicle coordinated search:

- Using *k* vehicles (heterogeneous – some air-based, some ground-based), conduct a coordinated search/sweep of a complex 2D/3D domain or terrain.
  - Minimize makespan (total time to completion)
  - Constraints: possible "tethering" constraints between vehicles, for communication/coordination purposes, and to be able to perform a sweep to detect a mobile, possibly adversarial evader.
  - Uncertainty: A priori distribution of target(s) locations, updated (Bayes) as search progresses



(a) (b)