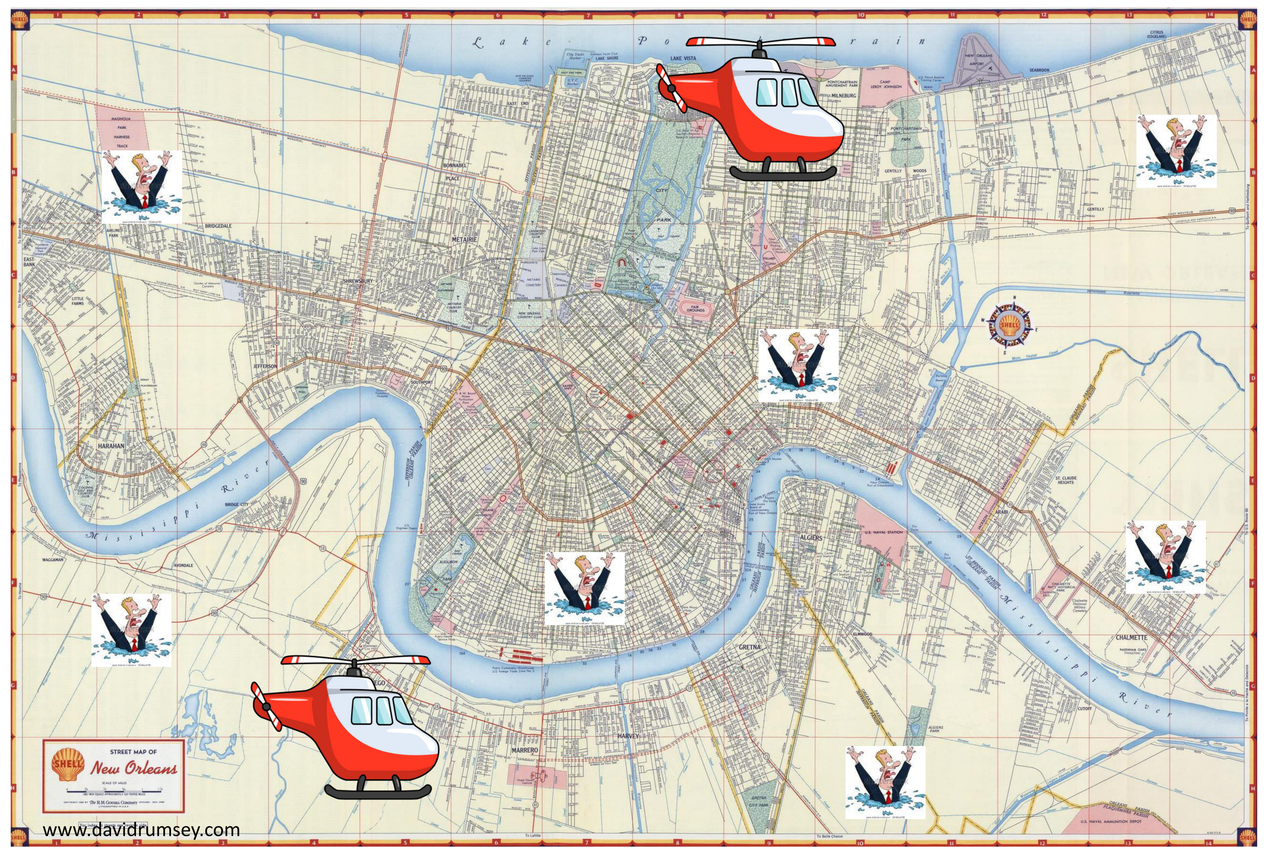
Motivation: A More Efficient Way to Emergency Evacuate and Transport

- In the wake of a natural disaster, with many left stranded and with no means of transportation, what is the most efficient path to get them to safety?
- Develop a system that:
  - Enables first responders to dispatch in an efficient way
  - Guides users with the optimal route of picking-up and delivering
  - Will save immense amounts of time, energy, and money!



Hurricane Harvey and Katrina

Hurricane Harvey and Katrina are tied for being the costliest tropical cyclones ever on record, devastating areas near the Gulf of Mexico. While the hurricanes were destructive, it was the ensuing floods that dealt the most damage. Hundreds of thousands of people were affected as their homes were destroyed and many had to evacuate the city.

Hurricane Harvey:

- Major roads were closed, constricting the flow of traffic out of the city
- 2.5 million people attempted to leave Houston, creating a huge traffic jam
- Dozens were killed by disruption of transportation during evacuation
- https://google.org/crisismap/2017-harvey

Hurricane Katrina:

- At least 1,833 died in the hurricane and subsequent floods
- Approximately 80% of the city was flooded
- Search and Rescue teams sent to evacuate thousands of people
- Huge population decrease (~50%) following aftermath, still hasn't recovered

Other Potential Applications

- Applicable to any situation involving optimizing transportation of people/goods

Military logistics

Package delivery system

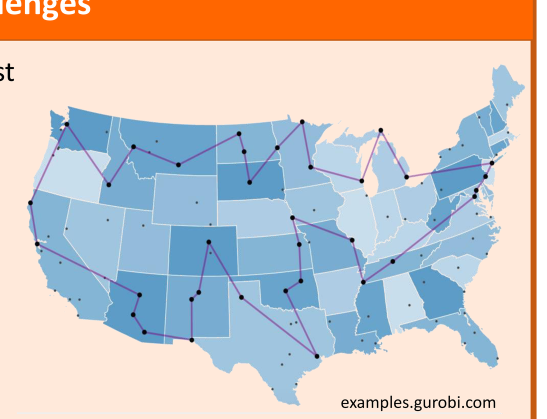
Ridesharing

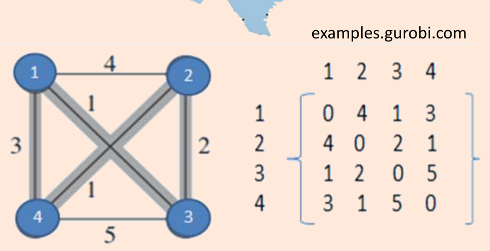
Questions

- How can I enable users to communicate and congregate in an efficient way? (information gathering)
- How can I generate optimal assignments of nodes to their initiators? (clustering)
- For each initiator, how can I come up with an optimal route to pick-up and deliver to assigned nodes? (routing search)

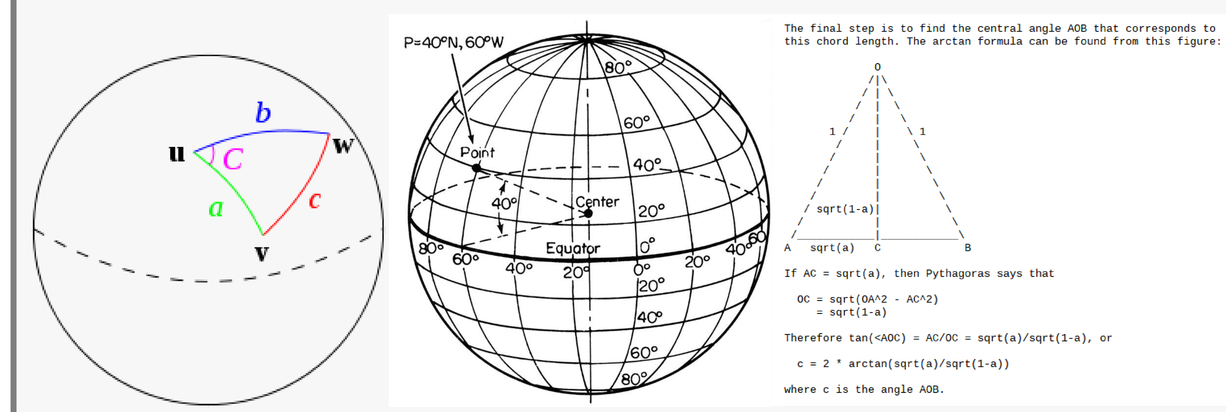
Objectives & Challenges

- Develop a system for efficiently calculate the shortest travel distance through numerous nodal points to a destination
- Travelling Salesman Problem (TSP)
  - TSP is an **NP-hard** problem
  - As the number of nodes increases, the time to compute increases exponentially ((N-1)!/2)
    - 5 nodes → 12 possible routes
    - 12 nodes → 19,958,400 possible routes
  - Due to this polynomial increase in calculation time, computer scientists have developed **heuristics** to approximate a **local optimum** close to the **global optimum**
  - Trading off a small portion of accuracy for a huge reduction in calculation time





Distance Matrix Calculation – Haversine Formula



Distance by Latitude and Longitude-Haversine Formula

- Because the problem requires computing distances over the face of the Earth, which is a sphere, need to account for the gradual slope of the Earth's surface over distance
  - To do this, we can use the Haversine Formula
- First find the measure of the arc between the two points from the center of the earth by using the Earth's radius.
- Able to find the distance by multiplying the circumference of the globe by the amount of degrees that makes up the arc between the points
- Initially use the Haversine Formula to find approximate distance for optimizing the carrier routes before converting the final result in Google Maps.

Initial Route (A "Good" Starting Solution) – Closest Path Algorithm

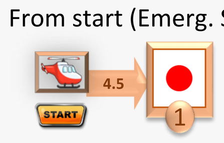
- Once the closest node to carrier path pairings have been made, generate a route to connect each carrier to their passengers and bring them to the final destination
  - Measure distance each carrier travels and add them up into a final sum of total distance

	START	Red	Blue	Yellow	Green	Flag
START	0	4.5	9	14.5	12	7.9
Red	4.5	0	6.6	16	8.7	7.2
Blue	9	6.6	0	19.5	11.1	4.6
Yellow	14.5	16	19.5	0	21.2	20.3
Green	12	8.7	11.1	21.2	0	10.2
Flag	7.9	7.2	4.6	20.3	10.2	0

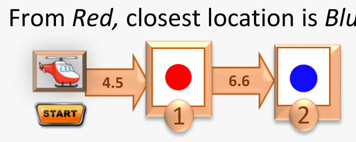
(From-to distance in miles)

Step 2a Initial Route: Build **initial route** by using "***closest path***" algorithm.

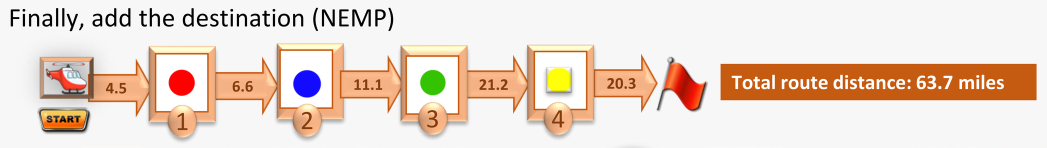
- From start (Emerg. Station), closest location is *Red (4.5 miles)*.

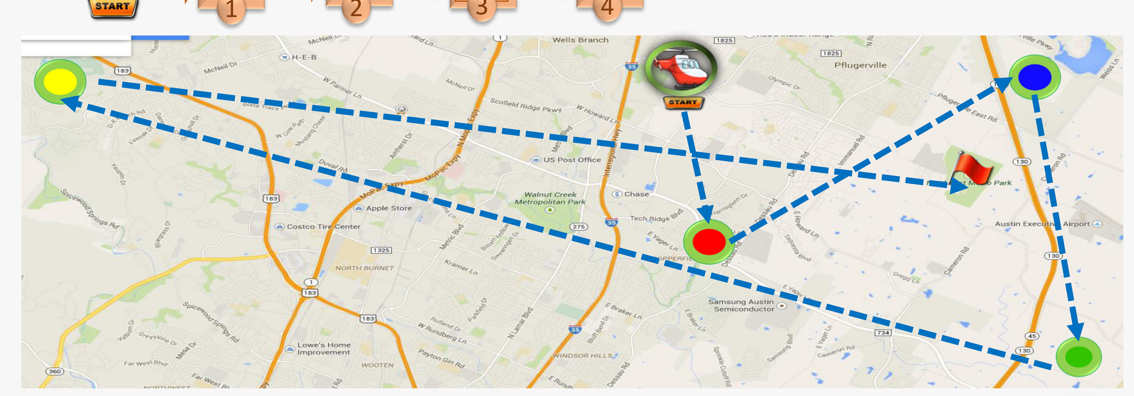


- From *Red*, closest location is *Blue (6.6 miles)*.



- From *Blue*, the closest location is *Green (11.1 miles)*, and from there next closest one is Yellow. Finally, add the destination (NEMP)

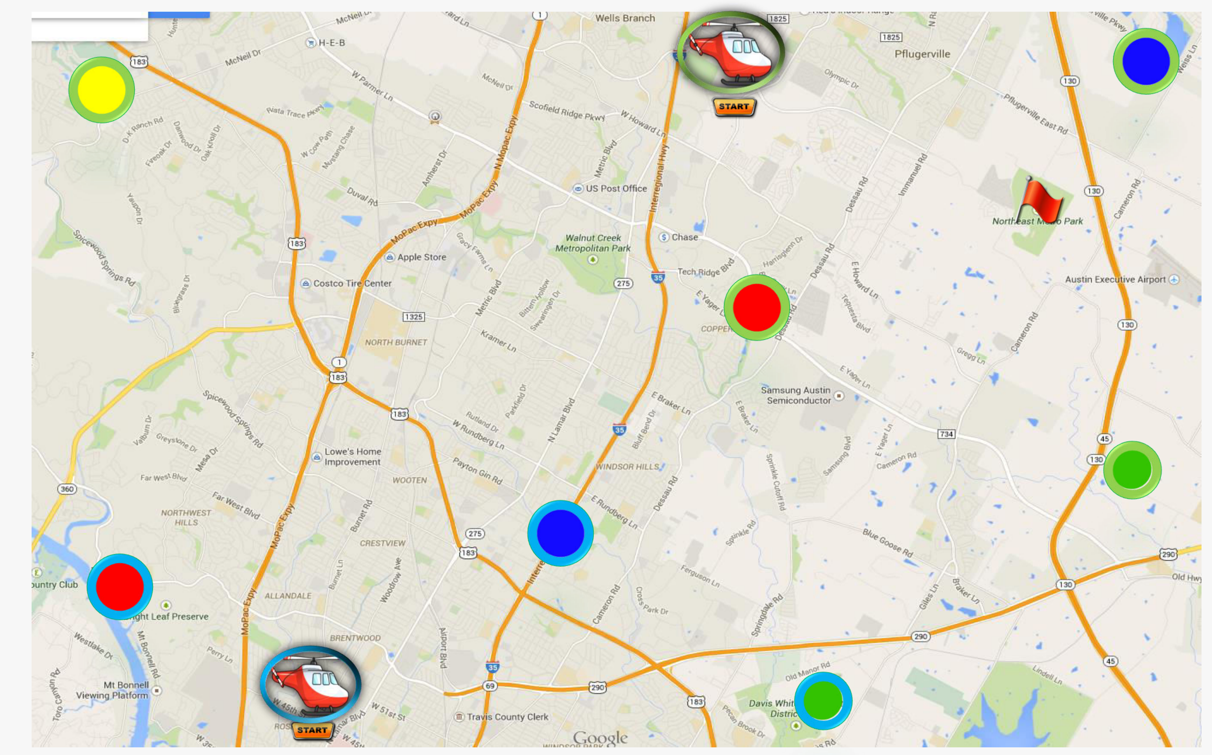




Node Allocation to Transport- Clustering Algorithm

- Calculate the distance between each carrier and the final destination to find the direct length between the two
- Pair the nodes with the closest path until carrier capacity has been filled OR all passengers have carriers
  - If all of the carriers' capacities have not been filled, then add more until all passengers have been accounted for

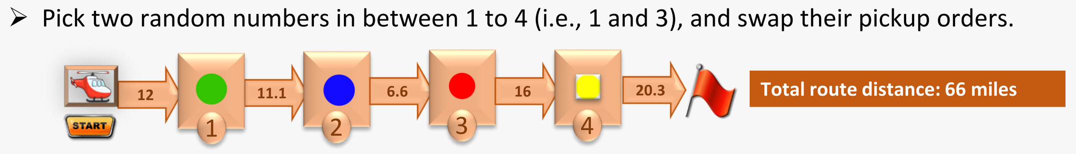
Step1 Clustering: By clustering algorithm; Blue, Yellow, Red and Green are assigned to Emerg. (D) (shown in green circles), and Red (2), Green (2), and Blue (2) are assigned to Emerg. (D2) (shown in blue circles).



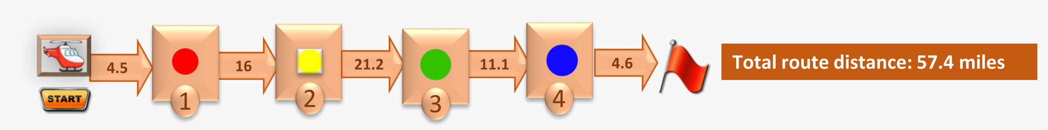
Now on, in Step 2, lets focus on the green team to provide optimal pickup route for Emerg.

Step 2b Optimize Route: Use search algorithms (GRASP w/ Simulated Annealing) to improve the initial solution.

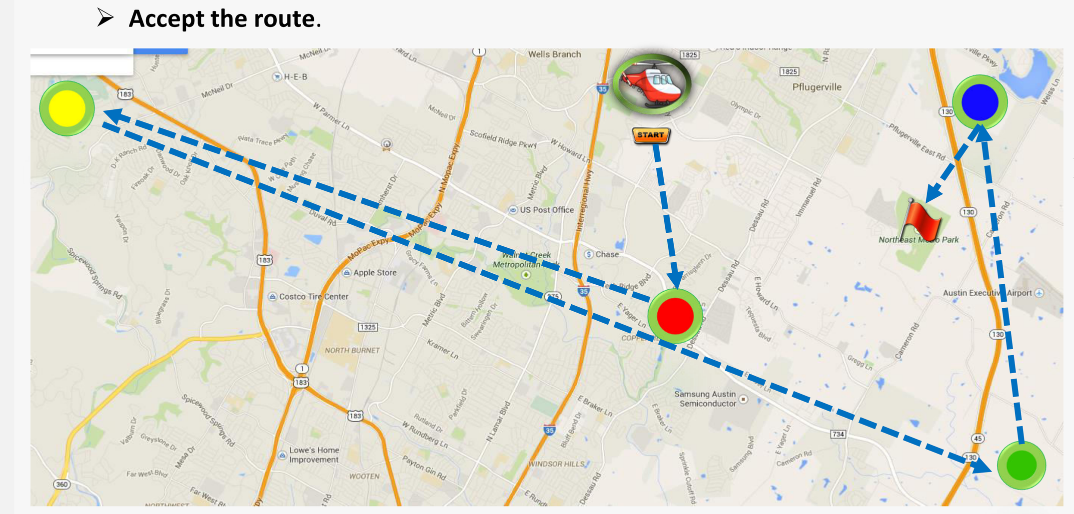
- Pick two random numbers in between 1 to 4 (i.e., 1 and 3), and swap their pickup orders.



- **Reject the route**, since total distance got worse (go back to the previous one with total distance of 63.7 miles).
- Pick two random numbers in between 1 to 4 (i.e., 2 and 4), and swap their pickup orders.

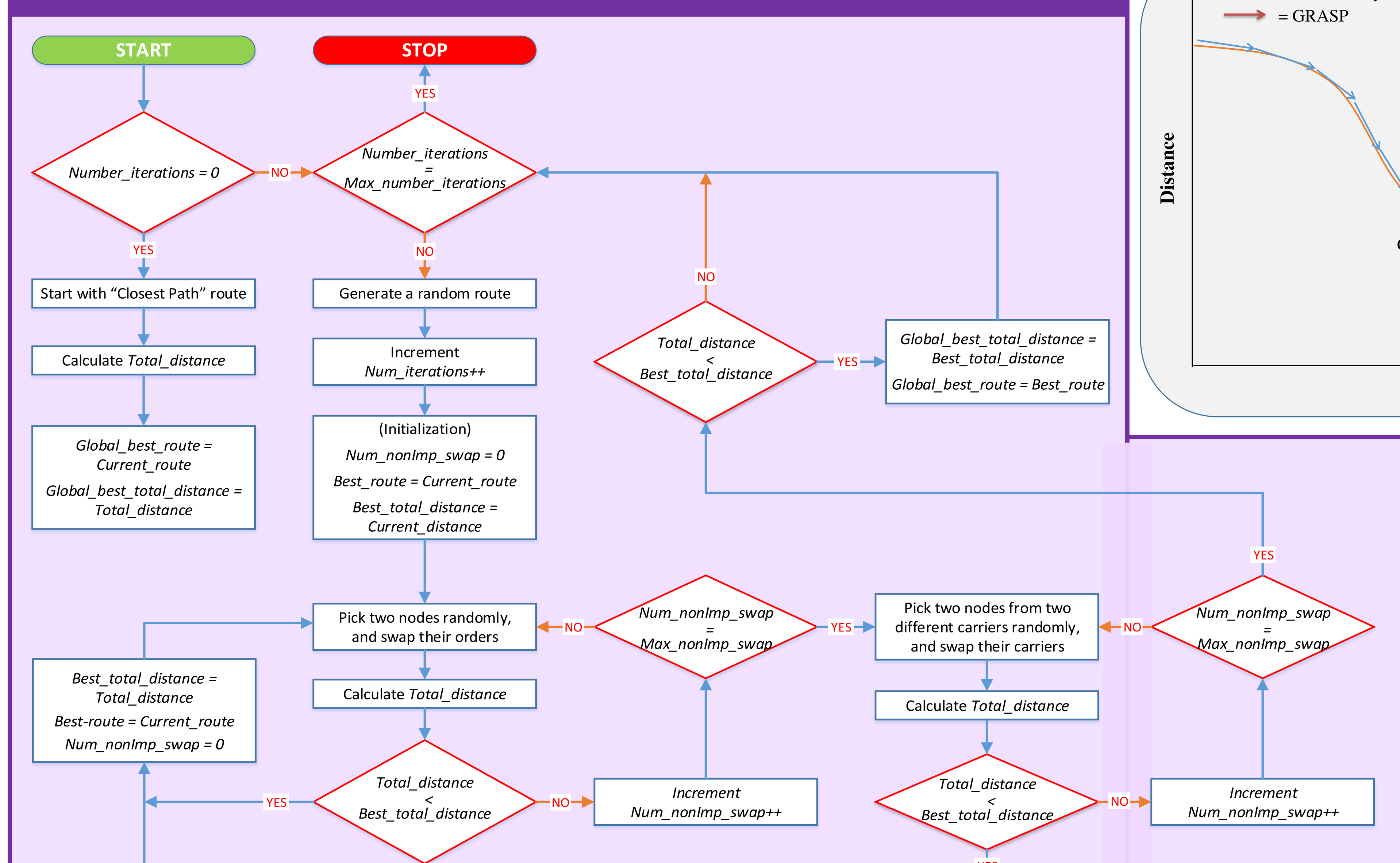


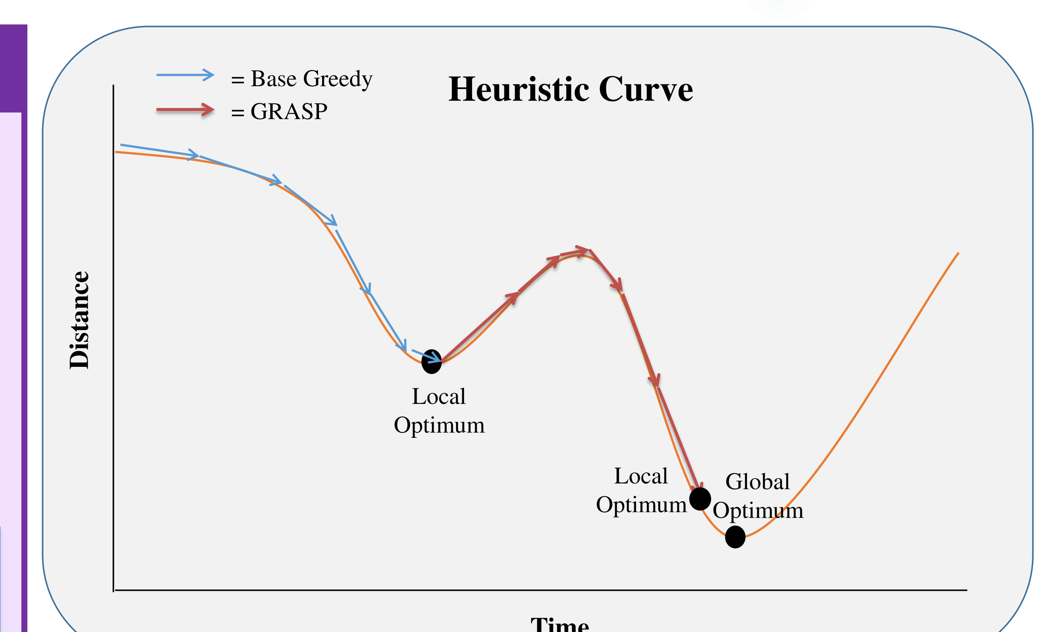
- **Accept the route.**

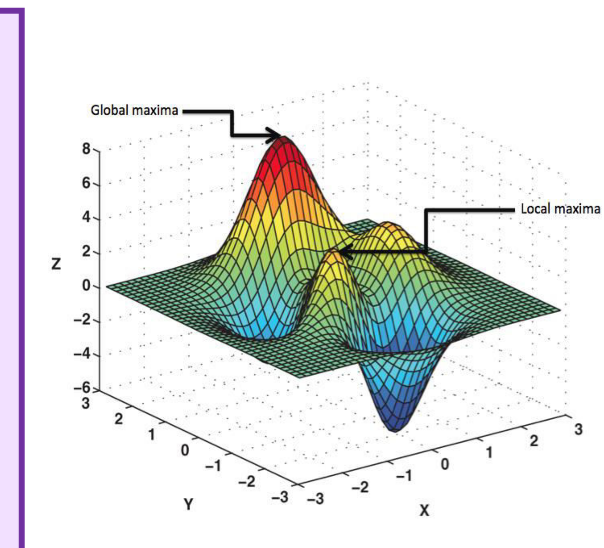


- Continue to search to reach optimal route. (see algorithm for details)

Optimized Route – Greedy Randomized Adaptive Search Procedure (GRASP)







Experimental Study

Data:

- Picked 50 random geographically diverse addresses as potential carrier and passenger locations.

Assumptions:

- Carrier has limited carry capacity
- Limited travel distance dictated by fuel capacity

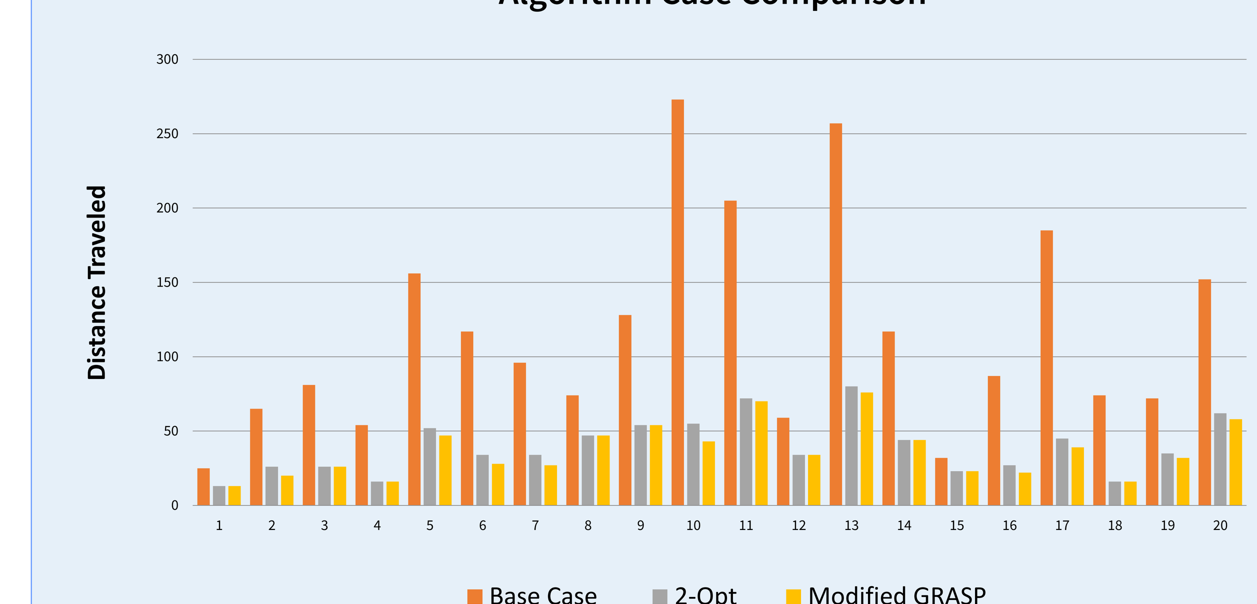
Methodology:

- To run diverse cases, generated 20 different scenarios with various numbers of passengers and carriers with changing pick-up capacities.
- Compared the output of GRASP algorithm to the base case AND the 2-Opt results

Summary of Cases

Case #	Base (Nearest Neighbor)	2-Opt Swap Greedy Search Heuristic	Modified GRASP Search Heuristic	Improvement % (Base Case to: 2-Opt / GRASP)	Saved Distance (miles)
1	25	13	13	48 %	12
2	65	26	20	60% / 69 %	39 / 45
3	81	26	26	68 %	55
4	54	16	16	70 %	38
5	156	52	47	~67 % / 70%	104 / 109
6	117	34	28	~71 % / 76%	83 / 89
7	96	34	27	65 % / 72%	62 / 69
8	74	47	47	36 %	27
9	128	54	54	58 %	74
10	273	55	43	~80 % / 84%	218 / 230
11	205	72	70	~65 % / 66%	133 / 135
12	59	34	34	~65 %	34
13	257	80	76	~69 % / 70%	177 / 181
14	117	44	44	~62 %	73
15	32	23	23	~28 %	9
16	87	27	22	~69% / 74%	60 / 65
17	185	45	39	~76% / 80%	140 / 146
18	74	16	16	~78 %	58
19	72	35	32	~51% / 56%	37 / 40
20	152	62	58	~59 % / 60%	90 / 94

Algorithm Case Comparison

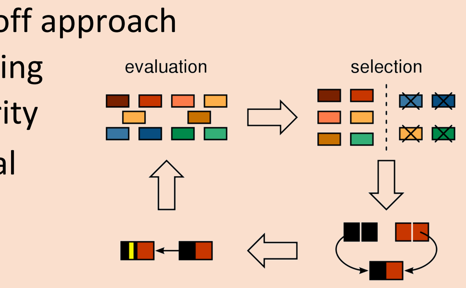


Results & Benefits

- **Quantitative savings:**
  - Average overall savings: **65% / 7.5%** (over Base/2-Opt)
  - Average distance savings: **79.84 mi. / 3 mi.** (over Base/2-Opt)
- **Qualitative savings:**
  - More lives saved due to shorter distance/time to travel
  - More resources able to be allocated for other purposes

Future Work

- Develop a fully automated system that requires no human intervention; it gathers information, generates the optimal assignment and routes, and then returns the assignments and routes back to the carriers and nodes.
  - Dynamic Rerouting system for a hands-off approach
  - Neural net integration for faster calculating
- Improve algorithm to take into account priority
- Look for other search algorithms for potential improvements in results
- Genetic heuristic algorithm
- Tabu-Search



Acknowledgements

- Dr. Mehmet Candas
- **Please see folder for more details of the references & research**