Route Optimization Algorithm

Indian Institute of Technology, Bombay

December 24, 2019

The Problem

- Single Depot Vehicle Scheduling Problem
- It is a combinatorial optimization problem for which finding the optimal solution is NP hard. Hence generally heuristic approaches are taken according to the size of data
- Constraints:
  - Time window
  - Bus capacity; limit and minimum 85% occupancy
  - Number of buses
- The objective is to minimise the operational cost of buses

Impact of the Problem

- The problem is important to be addressed because of its evident commercial and economical impact

Impact of the Problem

- The problem is important to be addressed because of its evident commercial and economical impact
- Owing to the large scale at which buses are used, even a 5% improvement can significantly reduce the carbon footprint

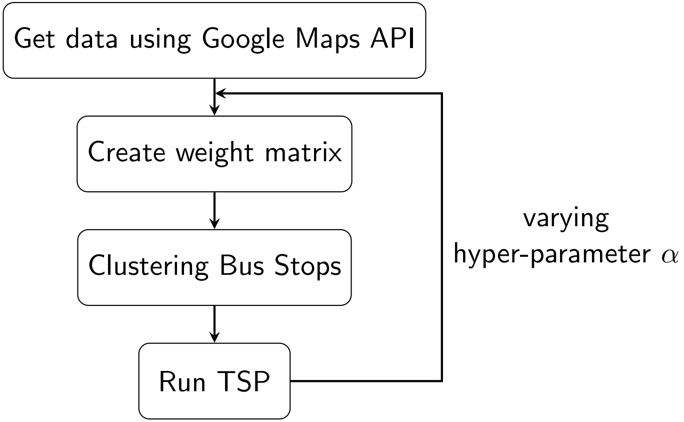
Impact of the Problem

- The problem is important to be addressed because of its evident commercial and economical impact
- Owing to the large scale at which buses are used, even a 5% improvement can significantly reduce the carbon footprint
- Solving this problem also helps in issues like Garbage Truck Scheduling and Goods Delivery Scheduling

Impact of the Problem

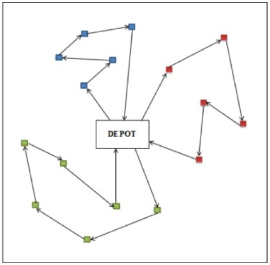
- The problem is important to be addressed because of its evident commercial and economical impact
- Owing to the large scale at which buses are used, even a 5% improvement can significantly reduce the carbon footprint
- Solving this problem also helps in issues like Garbage Truck Scheduling and Goods Delivery Scheduling
- This problem is generally approached using Integer Programming, by forming a set of equations and finding solutions and optimizing the process by using methods like ant colonization

Flow of Solving



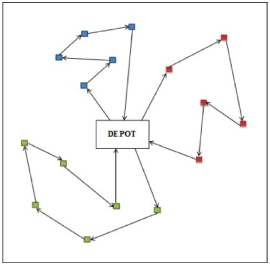
Travelling Salesman Problem(TSP)

- What is TSP and why TSP?



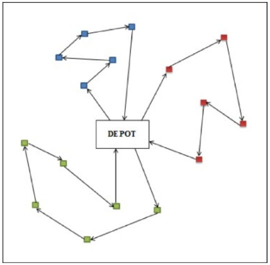
Travelling Salesman Problem(TSP)

- What is TSP and why TSP?
- Since its time complexity is O(n^2 2^n), we cannot use the algorithm directly as it is very inefficient



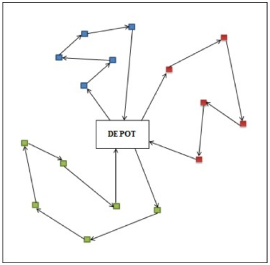
Travelling Salesman Problem(TSP)

- What is TSP and why TSP?
- Since its time complexity is O(n^2 2^n), we cannot use the algorithm directly as it is very inefficient
- We use clustering on the bus stops to improve the overall time complexity



Travelling Salesman Problem(TSP)

- What is TSP and why TSP?
- Since its time complexity is $O(n^2\ 2^n)$, we cannot use the algorithm directly as it is very inefficient
- We use clustering on the bus stops to improve the overall time complexity
- Once the bus stops are in clusters we will assign one bus to each cluster and use TSP to determine route of the bus



Why clustering improves time

- Assume ‘n’ nodes (bus stops)

Why clustering improves time

- Assume 'n' nodes (bus stops)
- If the n nodes are divided in k clusters, with n_i nodes in each cluster, such that $n_i \approx \mathrm{n}_j$ for all i,j, thus $kn_i = n$ Time complexity for TSP for each cluster is O(n_i^2 2^{n_i})

Why clustering improves time

- Assume ‘n’ nodes (bus stops)
- If the n nodes are divided in k clusters, with n_i nodes in each cluster, such that $n_i \approx$ n$_j$ for all i,j, thus $kn_i = n$ Time complexity for TSP for each cluster is O(n_i^2 2^{n_i})
- Thus, $T_{withoutclustering} \propto n^2 2^n$ and $T_{withclustering} \propto k n_i^2 2^{n_i}$ Therefore,

$$\frac{T_{withoutclustering}}{T_{withclustering}} = k.2^{n(1-1/k)}$$

Why clustering improves time

- Assume 'n' nodes (bus stops)
- If the n nodes are divided in k clusters, with n_i nodes in each cluster, such that $n_i \approx$ n$_j$ for all i,j, thus $kn_i = n$ Time complexity for TSP for each cluster is O(n_i^2 2^{n_i})
- Thus, $T_{withoutclustering} \propto n^2 2^n$ and $T_{withclustering} \propto kn_i^2 2^{n_i}$ Therefore,

$$\frac{T_{withoutclustering}}{T_{withclustering}} = k.2^{n(1-1/k)}$$

- For a fixed n, $T_{withoutclustering}$ is constant. As number of clusters k increases, $k.2^{n(1-1/k)}$ increases exponentially, thus $T_{withclustering}$ decreases

Why clustering improves time

- Assume 'n' nodes (bus stops)
- If the n nodes are divided in k clusters, with n_i nodes in each cluster, such that $n_i \approx$ n$_j$ for all i,j, thus $kn_i = n$ Time complexity for TSP for each cluster is O(n_i^2 2^{n_i})
- Thus, $T_{withoutclustering} \propto n^2 2^n$ and $T_{withclustering} \propto kn_i^2 2^{n_i}$ Therefore,

$$\frac{T_{withoutclustering}}{T_{withclustering}} = k.2^{n(1-1/k)}$$

- For a fixed n, $T_{withoutclustering}$ is constant. As number of clusters k increases, $k.2^{n(1-1/k)}$ increases exponentially, thus $T_{withclustering}$ decreases
- Hence with increase in number of clusters k, the time taken for TSP decreases exponentially

Clustering

- K-Means clustering has been used

Clustering

- K-Means clustering has been used
- Since each bus is going to cover all bus stops in a cluster, the number of stops in each cluster must satisfy the size constraints of the bus

Clustering

- K-Means clustering has been used
- Since each bus is going to cover all bus stops in a cluster, the number of stops in each cluster must satisfy the size constraints of the bus
- Number of clusters should not exceed the number of buses available

Clustering

- K-Means clustering has been used
- Since each bus is going to cover all bus stops in a cluster, the number of stops in each cluster must satisfy the size constraints of the bus
- Number of clusters should not exceed the number of buses available
- For number of clusters k, we find the smallest integer k satisfying:

$$k > (\text{number of people})/(\text{bus capacity})$$

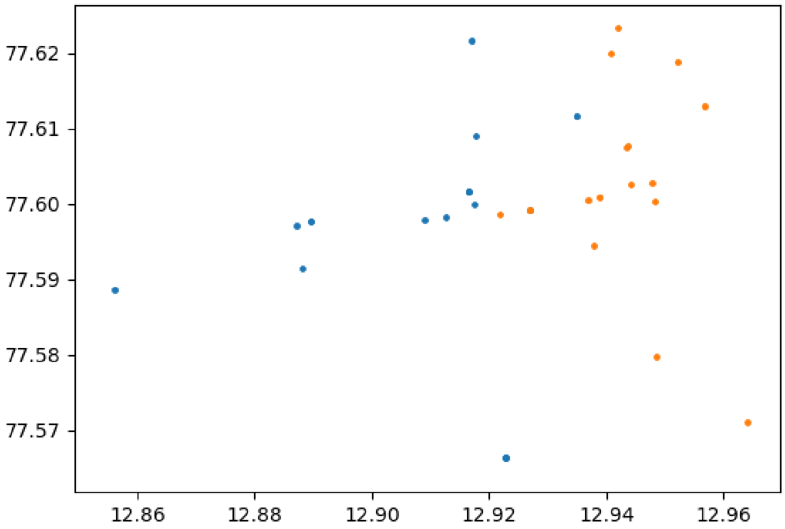
Clustering

- K-Means clustering has been used
- Since each bus is going to cover all bus stops in a cluster, the number of stops in each cluster must satisfy the size constraints of the bus
- Number of clusters should not exceed the number of buses available
- For number of clusters k, we find the smallest integer k satisfying:

$$k > (\text{number of people})/(\text{bus capacity})$$

- If the size of a cluster exceeds bus capacity then treat this cluster as an independent problem and use the procedure recursively over it

Clustering of the newly given data



Handling the Time Window constraint

- The TSP used previously considered only distances between the stops and not time

Handling the Time Window constraint

- The TSP used previously considered only distances between the stops and not time
- We replace the distances with a weighted average of distance between the stops and the time required to go from one stop to the other

$$weight(i,j) = \alpha \times distance(i,j) + (1-\alpha) \times time(i,j) \times v$$

Where v = average speed of the bus

Handling the Time Window constraint

- The TSP used previously considered only distances between the stops and not time
- We replace the distances with a weighted average of distance between the stops and the time required to go from one stop to the other

$$weight(i,j) = \alpha \times distance(i,j) + (1-\alpha) \times time(i,j) \times v$$

  Where v = average speed of the bus

- Run the TSP using these weights for a range of values of α

Handling the Time Window constraint

- The TSP used previously considered only distances between the stops and not time
- We replace the distances with a weighted average of distance between the stops and the time required to go from one stop to the other

$$\mathit{weight}(i,j) = \alpha \times \mathit{distance}(i,j) + (1-\alpha) \times \mathit{time}(i,j) \times v$$

  Where v = average speed of the bus
- Run the TSP using these weights for a range of values of α
- $\alpha = 1$ will give the minimum distance solution ignoring time window, $\alpha = 0$ will give the minimum time solution ignoring cost

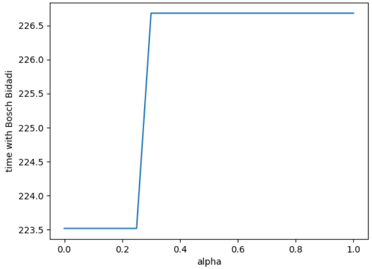
Handling the Time Window constraint

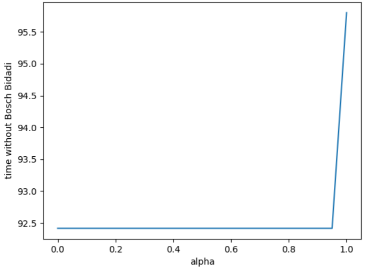
- The TSP used previously considered only distances between the stops and not time
- We replace the distances with a weighted average of distance between the stops and the time required to go from one stop to the other

$$\textit{weight}(i,j) = \alpha \times \textit{distance}(i,j) + (1-\alpha) \times \textit{time}(i,j) \times v$$

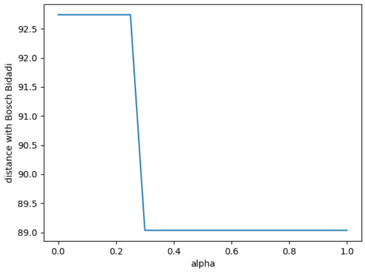
  Where v = average speed of the bus
- Run the TSP using these weights for a range of values of α
- $\alpha = 1$ will give the minimum distance solution ignoring time window, $\alpha = 0$ will give the minimum time solution ignoring cost
- We want to have a solution such that the time taken by the buses falls within the time window. We will use a particular value for α

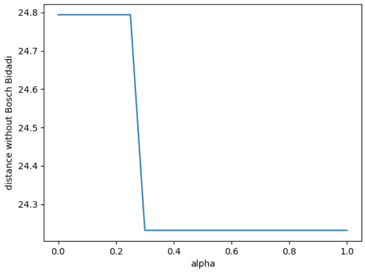
Handling the Time Window constraint





Handling the Distance constraint





Handling Real Time Traffic

- Mark the points which have been covered as visited

Handling Real Time Traffic

- Mark the points which have been covered as visited
- Schedule a Cron job for every 15 mins(say) which makes API calls to get distance and time matrix for only those points which are unvisited and update the weight matrix

Handling Real Time Traffic

- Mark the points which have been covered as visited
- Schedule a Cron job for every 15 mins(say) which makes API calls to get distance and time matrix for only those points which are unvisited and update the weight matrix
- Clustering remains same

Handling Real Time Traffic

- Mark the points which have been covered as visited
- Schedule a Cron job for every 15 mins(say) which makes API calls to get distance and time matrix for only those points which are unvisited and update the weight matrix
- Clustering remains same
- For each cluster find path of minimum weight covering all nodes where begin and end points are not the same

Handling Real Time Traffic

- Mark the points which have been covered as visited
- Schedule a Cron job for every 15 mins(say) which makes API calls to get distance and time matrix for only those points which are unvisited and update the weight matrix
- Clustering remains same
- For each cluster find path of minimum weight covering all nodes where begin and end points are not the same
- This is solved in the way same as that for TSP with a modified equation for Dynamic Programming

Handling Real Time Traffic

- Mark the points which have been covered as visited
- Schedule a Cron job for every 15 mins(say) which makes API calls to get distance and time matrix for only those points which are unvisited and update the weight matrix
- Clustering remains same
- For each cluster find path of minimum weight covering all nodes where begin and end points are not the same
- This is solved in the way same as that for TSP with a modified equation for Dynamic Programming
- The time for this computation will be lesser than that of the TSP done initially as number of points have reduced and hence also efficient

Results

- In the given data set, the number of people boarding is 95

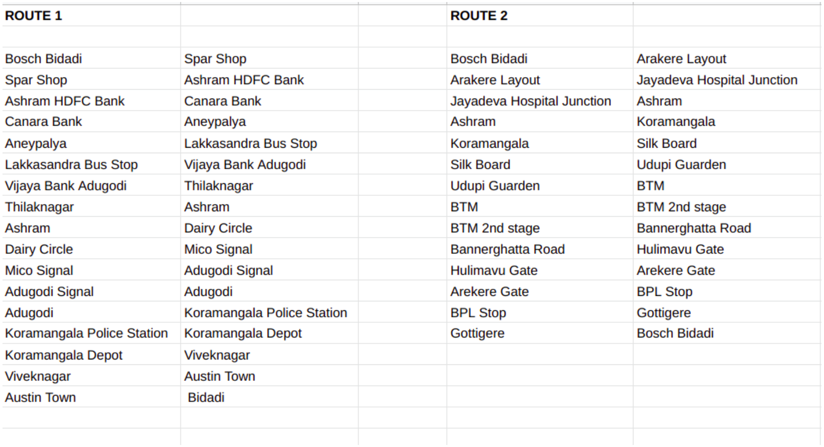
Results

- In the given data set, the number of people boarding is 95
- Since bus capacity is 50 and number of people 95, it is optimum to use 2 buses, 1 bus doesn't work, more than 2 buses will lead to atleast one bus with occupancy less than 85%

Results

- In the given data set, the number of people boarding is 95
- Since bus capacity is 50 and number of people 95, it is optimum to use 2 buses, 1 bus doesn't work, more than 2 buses will lead to atleast one bus with occupancy less than 85%
- An optimal solution is to use 2 buses following the routes :

Results

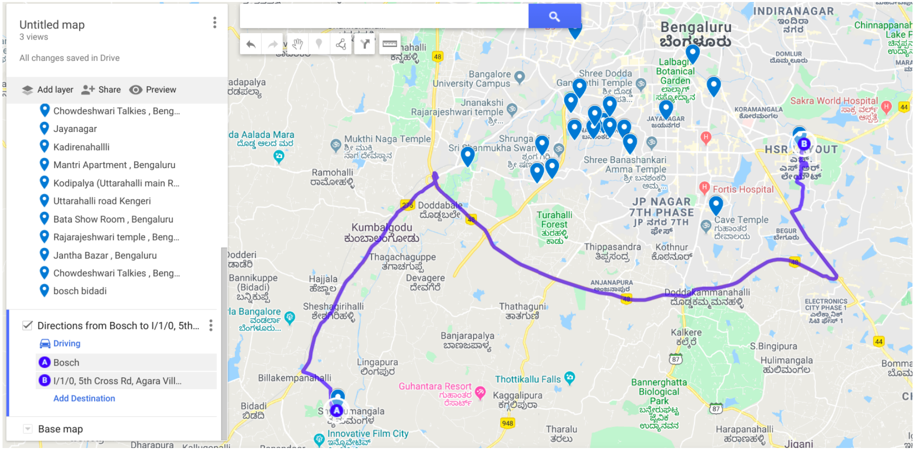


ROUTE 1		ROUTE 2	
Bosch Bidadi	Spar Shop	Bosch Bidadi	Arakere Layout
Spar Shop	Ashram HDFC Bank	Arakere Layout	Jayadeva Hospital Junction
Ashram HDFC Bank	Canara Bank	Jayadeva Hospital Junction	Ashram
Canara Bank	Aneypalya	Ashram	Koramangala
Aneypalya	Lakkasandra Bus Stop	Koramangala	Silk Board
Lakkasandra Bus Stop	Vijaya Bank Adugodi	Silk Board	Udupi Guarden
Vijaya Bank Adugodi	Thilaknagar	Udupi Guarden	BTM
Thilaknagar	Ashram	BTM	BTM 2nd stage
Ashram	Dairy Circle	BTM 2nd stage	Bannerghatta Road
Dairy Circle	Mico Signal	Bannerghatta Road	Hulimavu Gate
Mico Signal	Adugodi Signal	Hulimavu Gate	Arekere Gate
Adugodi Signal	Adugodi	Arekere Gate	BPL Stop
Adugodi	Koramangala Police Station	BPL Stop	Gottigere
Koramangala Police Station	Koramangala Depot	Gottigere	Bosch Bidadi
Koramangala Depot	Viveknagar		
Viveknagar	Austin Town		
Austin Town	Bidadi		

Results

- For the given sample data set, no solution is possible which satisfies all the given constraints
- Reason is that the depot 'Bosch Bidadi' is itself at a far off distance from the other stops, explained on maps
- 'Mantri Apartment' is itself 50 km and at a time of 1 hr from the depot, while time window is of 1hr 20 min
- We can handle such outliers
- Since number of people is 29 and 85% of bus capacity is 27, only one bus can be used. Solution for this leads to a minimum time of 4 hrs which is not acceptable
- Possible way to reduce time is to use 2 or more buses but then they will have less occupancy

Results


Untitled map
3 views
All changes saved in Drive
Add layer Share Preview
- Chowdeshwari Talkies , Beng...
- Jayanagar
- Kadirenahallli
- Mantri Apartment , Bengaluru
- Kodipalya (Uttarahalli main R...
- Uttarahalli road Kengeri
- Bata Show Room , Bengaluru
- Rajarajeshwari temple , Beng...
- Jantha Bazar , Bengaluru
- Chowdeshwari Talkies , Beng...
- bosch bidadi

Directions from Bosch to I/1/0, 5th...
Driving
A Bosch
B I/1/0, 5th Cross Rd, Agara Vill...
Add Destination
Base map

Challenges faced

- While using Google Maps APIs to get distances and travel time between places, some places were causing ambiguities, eg. for Jantha Bazar, the location being returned was over 1000km from Bosch Bidadi

Challenges faced

- While using Google Maps APIs to get distances and travel time between places, some places were causing ambiguities, eg. for Jantha Bazar, the location being returned was over 1000km from Bosch Bidadi
- To handle this, whenever there are such locations which have an average distance more than 100 km from the other points, we append the city name to it for the API call

Quantum Algorithm to solve the TSP

- NP hard problem in combinatorial optimization takes exponential time order for solving by classical brute force method

[1] *Karthik Srinivasan, Saipriya Satyajit, Bikash K Behera, and Prasanta K. Panigrahi*, **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**

Quantum Algorithm to solve the TSP

- NP hard problem in combinatorial optimization takes exponential time order for solving by classical brute force method
- Quantum Algorithm solves the TSP using phase estimation technique. Distance between points is encoded as phases

[1] *Karthik Srinivasan, Saipriya Satyajit, Bikash K Behera, and Prasanta K. Panigrahi*, **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**

Quantum Algorithm to solve the TSP

- NP hard problem in combinatorial optimization takes exponential time order for solving by classical brute force method
- Quantum Algorithm solves the TSP using phase estimation technique. Distance between points is encoded as phases
- Construct unitary operators whose eigenvectors are the computational basis states and eigenvalues are various combinations of these phases

[1] *Karthik Srinivasan, Saipriya Satyajit, Bikash K Behera, and Prasanta K. Panigrahi*, **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**

Quantum Algorithm to solve the TSP

- NP hard problem in combinatorial optimization takes exponential time order for solving by classical brute force method
- Quantum Algorithm solves the TSP using phase estimation technique. Distance between points is encoded as phases
- Construct unitary operators whose eigenvectors are the computational basis states and eigenvalues are various combinations of these phases
- Then we apply phase estimation algorithm to certain eigenstates which give us all the total distances possible for all the routes

[1] *Karthik Srinivasan, Saipriya Satyajit, Bikash K Behera, and Prasanta K. Panigrahi*, **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**

Quantum Algorithm to solve the TSP

- NP hard problem in combinatorial optimization takes exponential time order for solving by classical brute force method
- Quantum Algorithm solves the TSP using phase estimation technique. Distance between points is encoded as phases
- Construct unitary operators whose eigenvectors are the computational basis states and eigenvalues are various combinations of these phases
- Then we apply phase estimation algorithm to certain eigenstates which give us all the total distances possible for all the routes
- After obtaining the distances we can search through this information using the quantum search algorithm for finding the minimum to find the least possible distance as well the route taken

[1] *Karthik Srinivasan, Saipriya Satyajit, Bikash K Behera, and Prasanta K. Panigrahi*, **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**

Quantum Algorithm to solve the TSP

- NP hard problem in combinatorial optimization takes exponential time order for solving by classical brute force method
- Quantum Algorithm solves the TSP using phase estimation technique. Distance between points is encoded as phases
- Construct unitary operators whose eigenvectors are the computational basis states and eigenvalues are various combinations of these phases
- Then we apply phase estimation algorithm to certain eigenstates which give us all the total distances possible for all the routes
- After obtaining the distances we can search through this information using the quantum search algorithm for finding the minimum to find the least possible distance as well the route taken
- This provides us a quadratic speedup over the classical brute force method for a large number of cities [1]

[1] *Karthik Srinivasan, Saipriya Satyajit, Bikash K Behera, and Prasanta K. Panigrahi*, **Efficient quantum algorithm for solving travelling salesman problem: An IBM quantum experience**