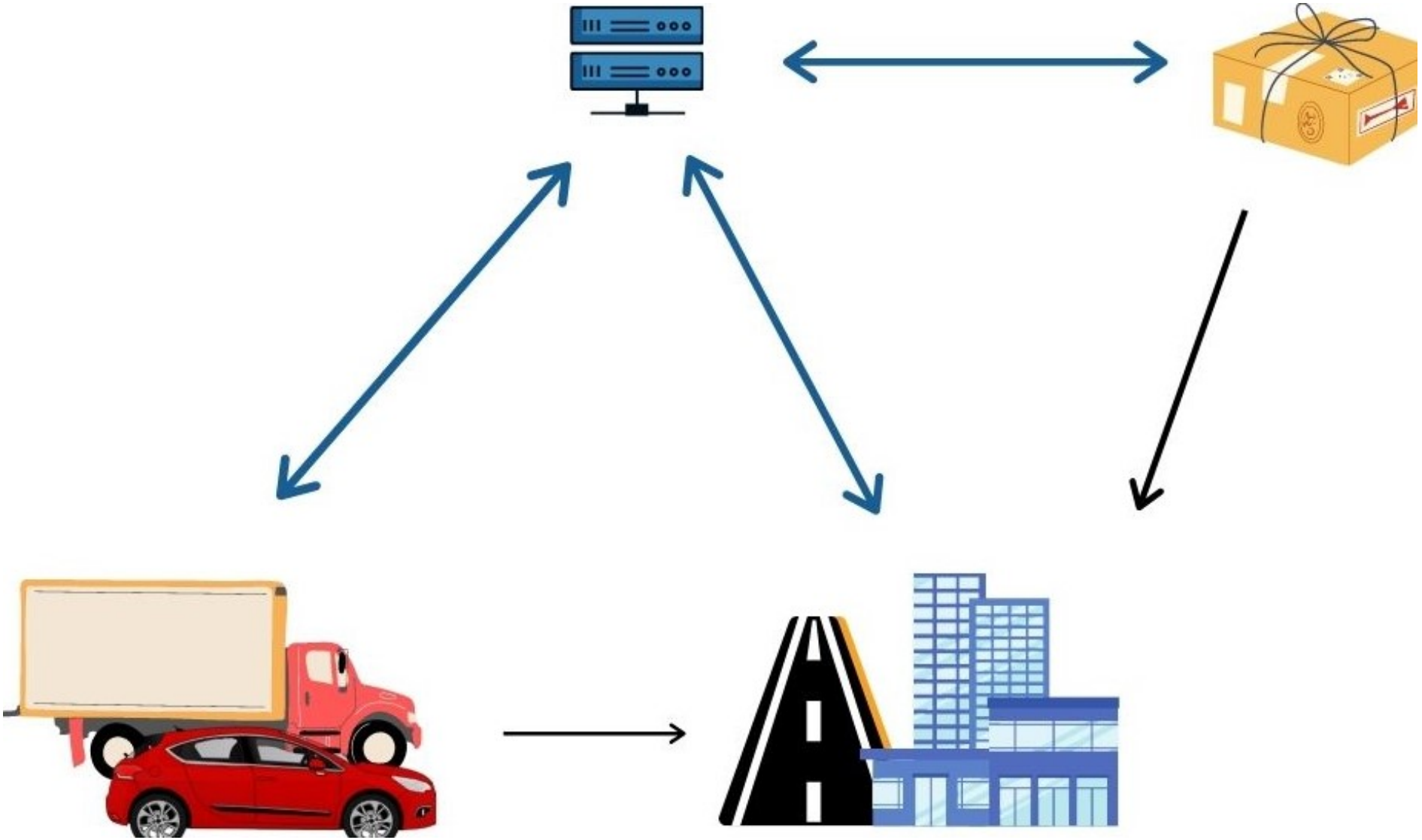
PROJECT LOAD-UNLOAD

New Approach to Supply Chains and Transportation of Goods using Search Algorithms

Nivedit Jain (B18CSE039)

October 2020

Introduction



Importance

- reduce the cost of transportation of goods
- support MSMEs (Micro Small and Medium Enterprises)
- effective for eCommerce businesses
- reduce the travel cost of people travelling by road.
- potentially increase the income for various drivers, like truck drivers
- reduce hundreds of thousands of good carriers from the road
- solving the problem of congestion on-road
- positively affecting our environment in a number of ways
- connecting remote locations

Problem Formulation

A vehicle has following parameters

- Start Location (A)
- End Location (B)
- Maximum more weight (V_m)
- Maximum more volume (V_v)

Problem Formulation

- V_v and V_m could be decreased by the driver as per needs.
- We assume the vehicle has registered with following parameters in our system.
- Vehicle needs to come to the nearest pickup centre.

Problem Formulation

Each parcel has following parameters

- Weight (P_w)
- Volume (P_v)
- Destination (P_d)
- Price to deliver (P_m)

Problem Formulation

we assume the to driver $x\%$ of P_m is passed, so,

$$P_c = \frac{x \times P_m}{100} \tag{1}$$

Problem Formulation

cost function (cf) assuming all picked are delivered, could be defined as,

$$cf([A \ldots B], TIME, \{PC_1 \ldots PC_n\}) =$$
$$c_1 len([A \ldots B]) + c_2 TIME - c_3 \sum_{i=1}^{i=n} P_{C\,i} \tag{2}$$

where,

Problem Formulation

- $len([A \ldots B])$ denotes distance of path from A to B
- $TIME$ denotes the time of travel
- n is the total number of parcels assigned at beginning
- c_1, c_2 and c_3, denotes importance of distance, time and money received.

Here,

$$c_i > 0 \tag{3}$$

Problem Formulation

cost function (cf) assuming all picked might not have be delivered, could be defined as,

$$cf([A \ldots B], TIME, \{PC_1 \ldots PC_n\}, U) = \\ c_1 len([A \ldots B]) + c_2 TIME - c_3 \sum_{i=1}^{i=n} PC_i + m \times Size(U) \tag{4}$$

where,

Problem Formulation

- $Size(U)$ denotes the number of elements in set of undelivered parcels (from parcels assigned to vehicle)
- m is a very large number never achievable by cf with $Size(U) = 0$

Problem Formulation

$$TIME = \sum_{i=1}^{i=k} Time_i + m \times T_S + n \times L_S \tag{5}$$

Problem Formulation

Our goal, to find a optimal assignment of parcels and also route, such that cost function (cf) is minimized, under the following constraints,

$$\sum_{i=1}^{i=m} P_{w\,i} \leq V_M \tag{6}$$

$$\sum_{i=1}^{i=m} P_{v\,i} \leq V_V \tag{7}$$

State Definition

$$S = (City, \{P_1, \ldots, P_n\}) \tag{8}$$

State Definition

Special Start State,

$$S = (City_0, \{\}) \tag{9}$$

State Definition

Special Start State,

$$S = (B, \{P_1, \ldots, P_m\}) \tag{10}$$

Number of States

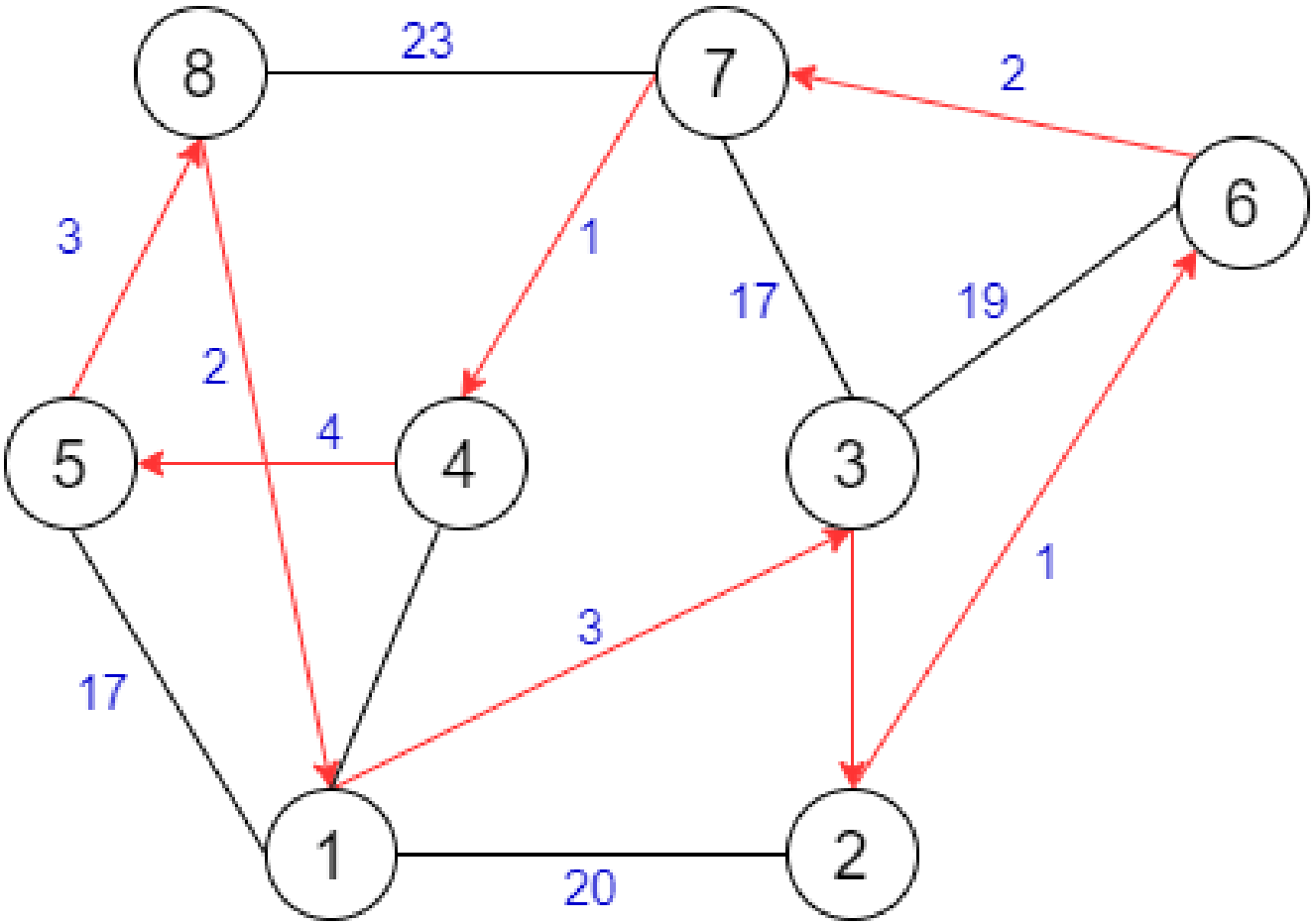
Number of possible parcel assignments, (Total P)

$$\binom{P}{0} + \binom{P}{1} + \cdots + \binom{P}{P} = 2^P \tag{11}$$

Let there be N cities, so,

$$Total\ States = N \times 2^P + 1 \tag{12}$$

Computational Hardness - TSP



Computational Hardness - TSP

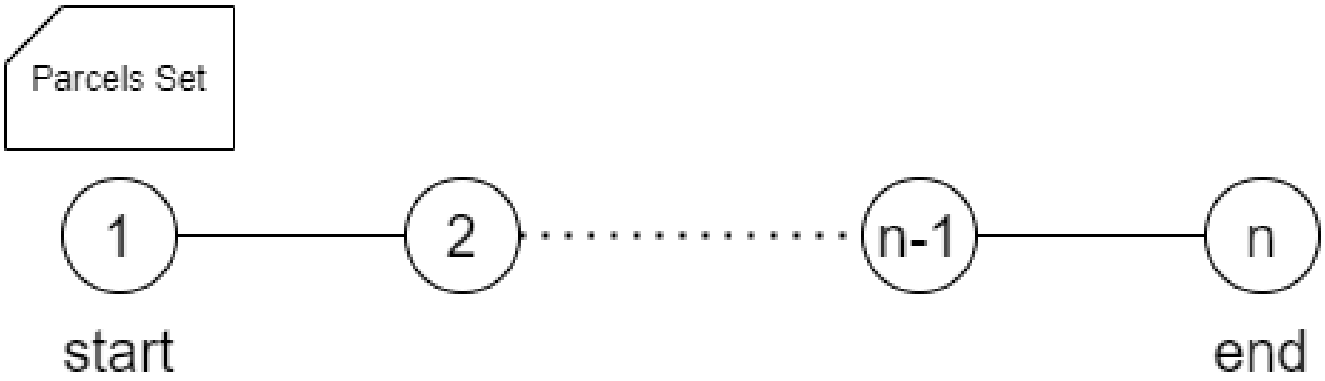
- Set Start Location to Destination.
- Set c_2 to 0, and set c_1 and c_3 to 1.
- Let our input graph has N cities, initially available parcels at start pickup center $N-1$, such that each parcel for a different city and follows following constraints,

$$\sum_{i=0}^{i=N} P_{Wi} \leq V_M \tag{13}$$

$$\sum_{i=0}^{i=N} P_{Vi} \leq V_V \tag{14}$$

$$\sum_{i=0}^{i=N} P_{Ci} \geq len([A \ldots B]) \tag{15}$$

Computational Hardness - 0-1 Knapsack

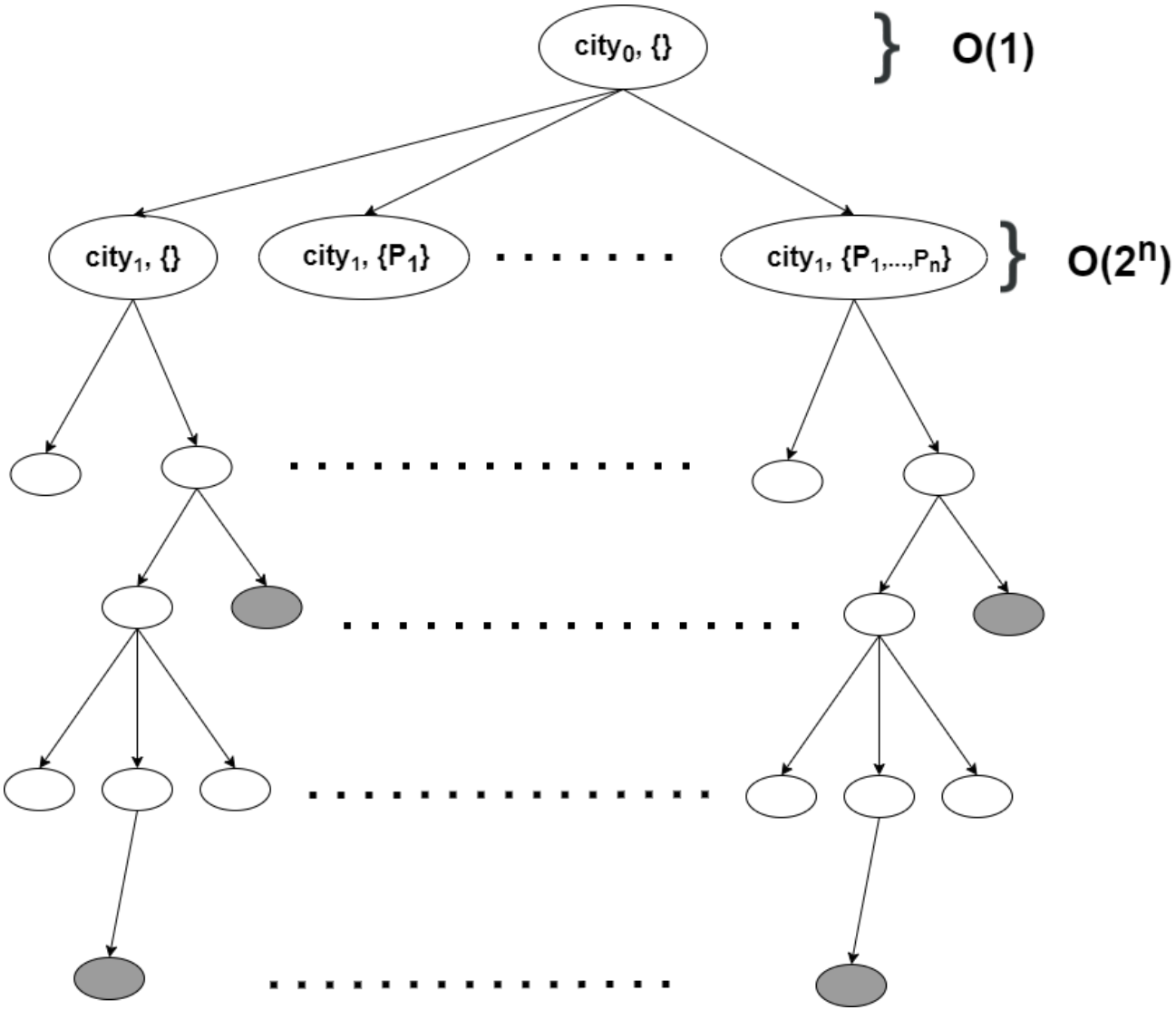


cities and road to be a Linear Graph with start state and goal state on two different ends, with c_1 and c_2 be 0 and c_3 be 1. We can clearly see most optimal solution would be one which highest valued packages in our vehicle under our constrains, which is a 0-1 Knapsack Problem.

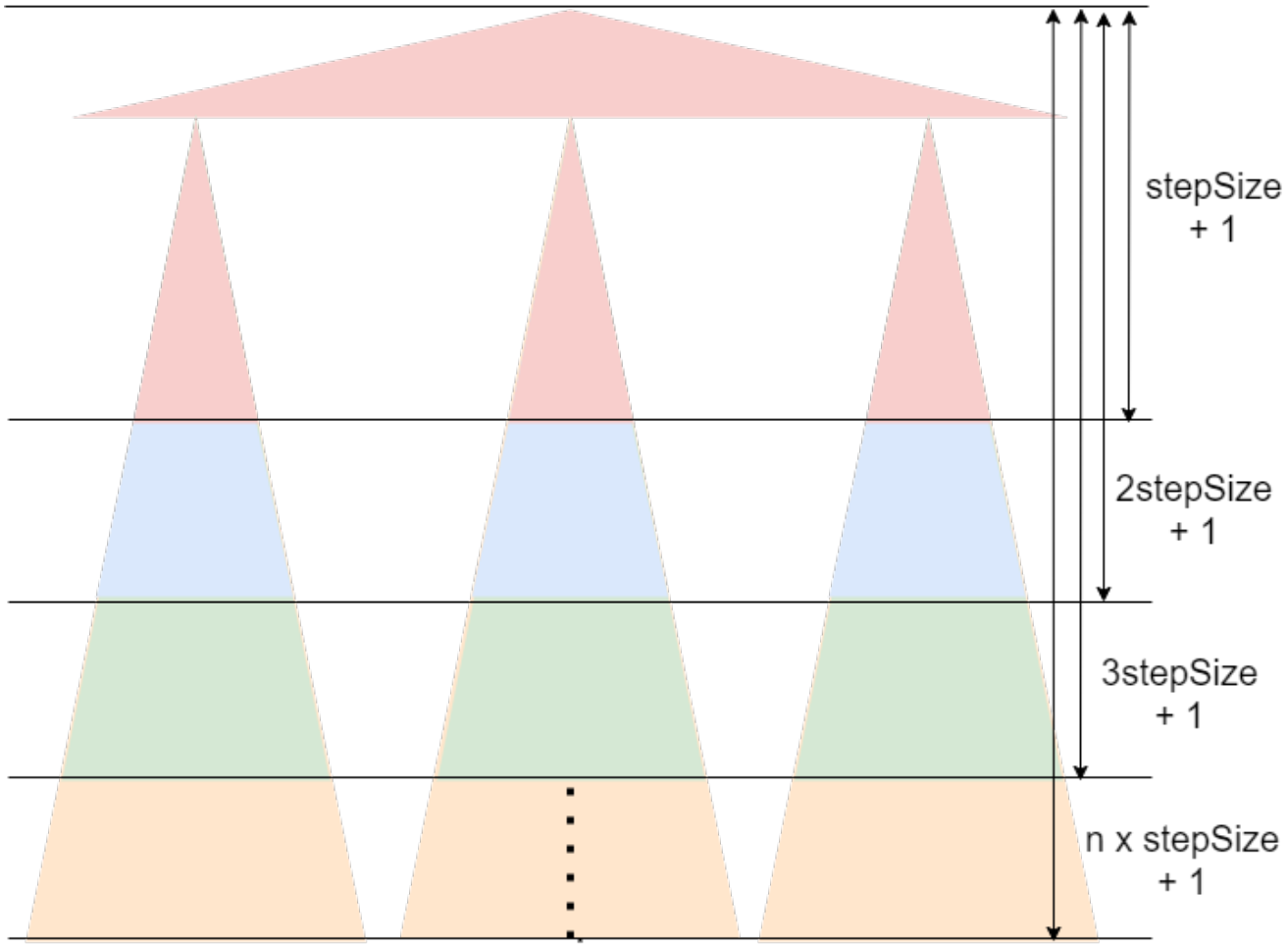
Computational Hardness

Solving our problem seems at least as hard as Travelling Salesman Problem and Knapsack Problem, our problem seems like a combination of both these NP-Hard Problems

Algorithm - Successor Function



Algorithm - IDDFS



Algorithm - IDDFS

- *stepSize*, the number of levels in depth we can go down in one step of iterative deepening, that means we will give the best solution till that level of depth.
- *epochs*, the number of times we go down, that means, we will output the best solution till $depth = stepSize \times epochNumber + 1$ (In the first *epoch* we go to $stepSize + 1$ depth, we do so to accommodate the special first state successor function.)

Algorithm - Time Complexity

time complexity would be asymptomatically equal to number of total number of states till that epoch (IDDFS has time complexity of B^M).Number of states visited,

$$\begin{aligned} &= 2^P \times \left(\frac{N^{epoch_i \times stepSize+1} - 1}{N-1}\right) \\ &= O(2^P \times N^{epoch_i \times stepSize}) \end{aligned} \tag{16}$$

Algorithm - Time Complexity

if we are running our algorithm for K epochs then running time would be

$$\begin{aligned} O(2^P \times N^{stepSize}) + O(2^P \times N^{2\times stepSize}) + \dots \\ \dots + O(2^P \times N^{K\times stepSize}) = O(2^P \times N^{K\times stepSize}) \end{aligned} \tag{17}$$

Algorithm - Space Complexity

Space Complexity (similar to IDDFS $B \times M$)

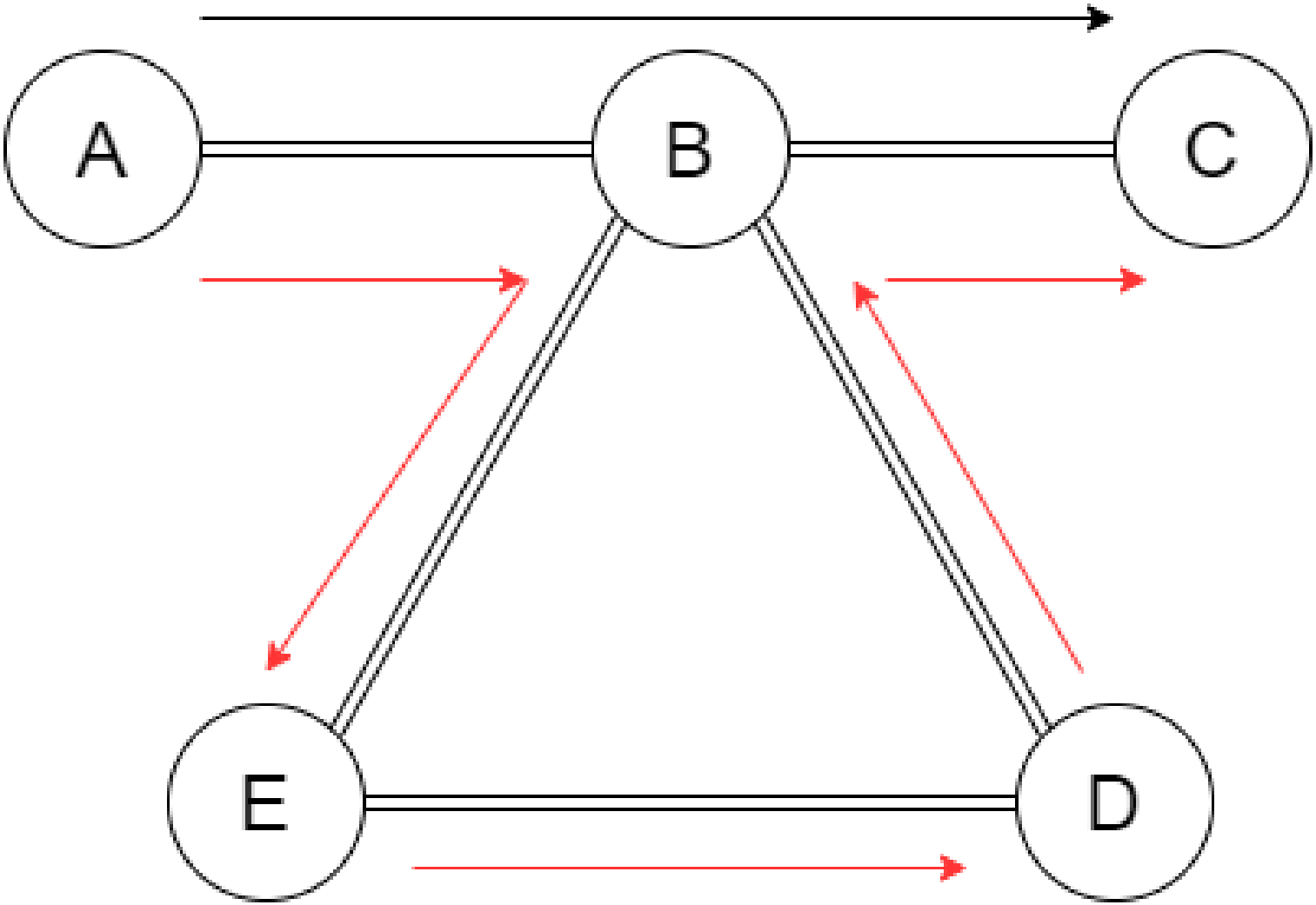
$$\begin{aligned} O(2^P + N \times (K \times stepSize) + K \times stepSize) \\ = O(2^P + N \times K \times stepSize) \end{aligned} \tag{18}$$

Algorithm - Claim

Claim

If there is a cycle in a graph then for optimal solution vehicle must take cycle once or do not take at all.

Algorithm - Claim



Algorithm - Claim

We cannot compare cf_0 and cf_1 without data, however, we can definitively that

$$cf_1 < cf_2 < \cdots < cf_n \tag{19}$$

Algorithm - Claim

Result - Travel each edge at most one time.

Algorithm - Claim

Claim

Maximum number of epochs that will lead to optimal solutions are bounded.

Algorithm - Claim

We know that for optimal solution be we need to travel each edge at max once, so, maximum possible depth is,

$$\frac{N \times (N-1)}{2} + 1 \tag{20}$$

as clique has maximum number of edges. So,

$$stepSize \times epoch_{max} + 1 = \frac{N \times (N-1)}{2} + 1 \tag{21}$$

which gives,

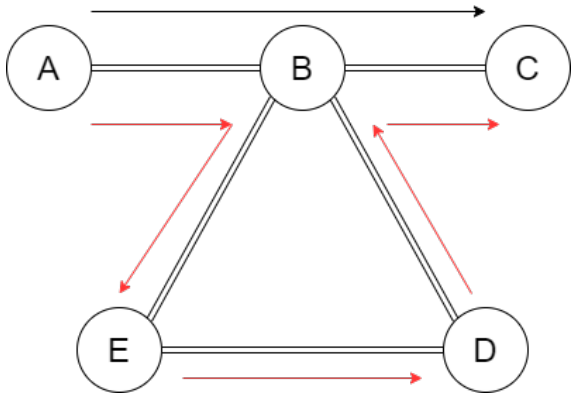
$$epoch_{max} = \lceil \frac{N \times (N-1)}{2 \times stepSize} \rceil \tag{22}$$

Algorithm - Pruning

we can use the following two conditions for edge pruning

- In a branch, we should not repeat the edge
- After first state check the feasibility of assignments and remove impossible assignments.

Algorithm - Example



start location be A and destination be C. For simplicity for now we have assumed c_1 and c_3 to be 1 and c_2 be 0. Let m be 1,00,000. V_v be 10 and V_M be 20. Let *stepSize* be 2. So $epochs_{max}$ become 3. We also assumed that at initial Pickup Centre following packets represented as (destination, volume, weight, P_C) :-: {P1 : (C, 5, 10, 100), P2 : (E, 5, 10, 50), P3 : (D, 7, 4, 140)} Let edges with weights (distance only for now) be (from, to, distance) (we are not taking time into consideration here as c_2 is 0) {(A, B, 1), (B, C, 2), (B, E, 1), (E, D, 1), (B, D, 2)}

Algorithm - Example - Epoch1

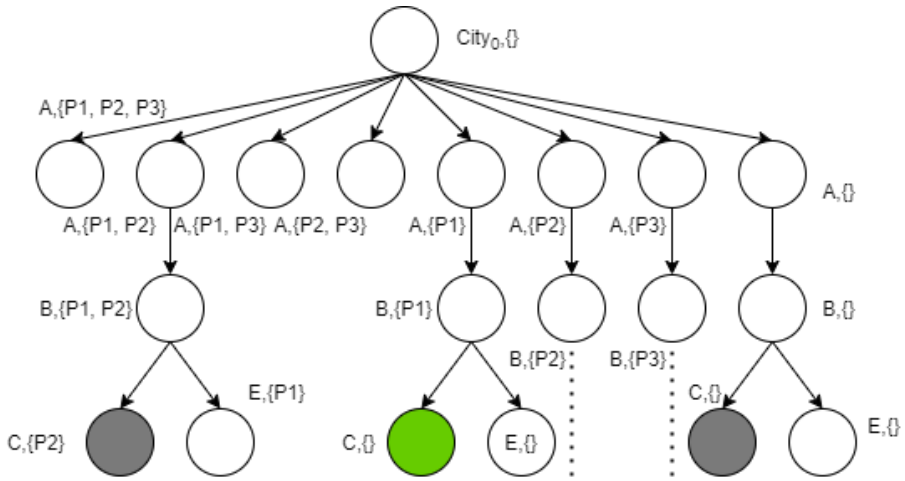


Figure: Search Graph after $epoch_1$ the best solution shown in green till here, with cost of, $(2+3)-(100)=-97$, we pruned the the nodes as per rules and dotted lines indicate that there will be more search but was unable to draw due to space constrains while drawing. We have not shown B-D as it will behave exactly same as B-E in this case

Algorithm - Example - Epoch2

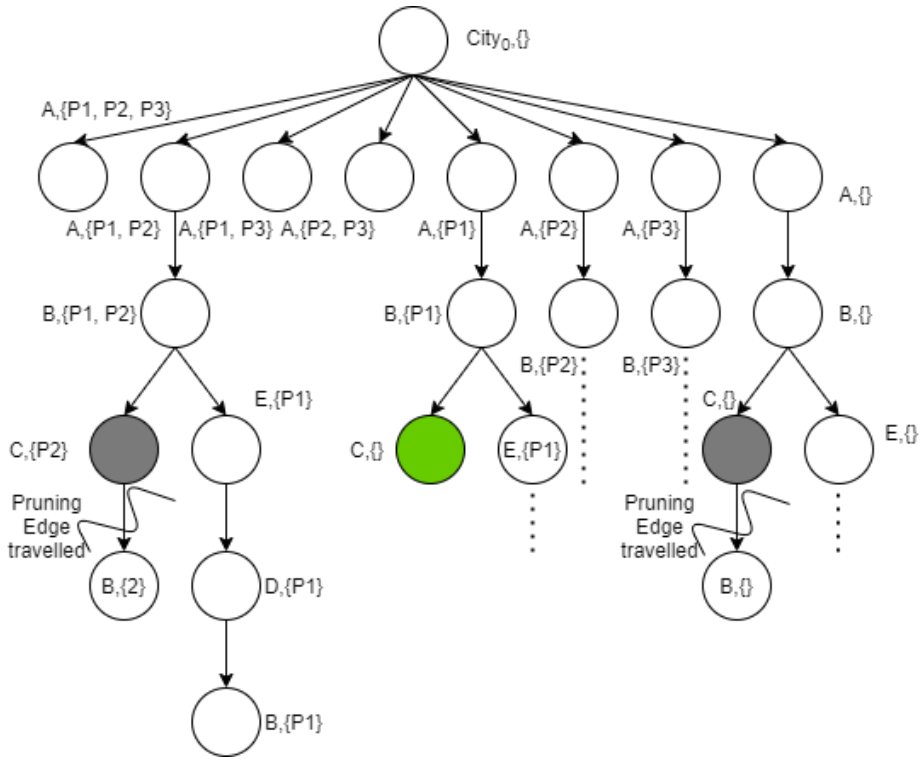


Figure: Search Graph after $epoch_2$ as we didn't reach a goal state in this epoch so best state remains the same.

Algorithm - Example - Epoch3

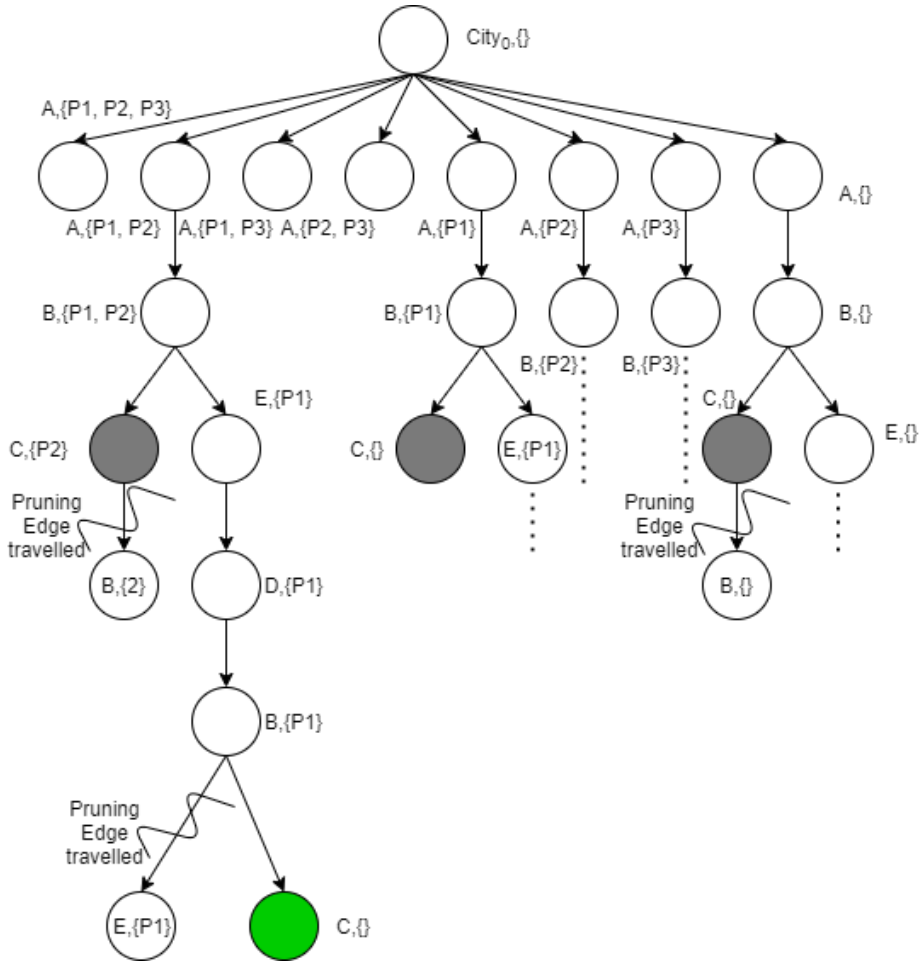


Figure: Search Graph after $epoch_3$, we are able to find a better state taking the bath $A \rightarrow B \rightarrow E \rightarrow D \rightarrow B \rightarrow C$, with packets { P1 , P2} with a cost of $(1 + 1 + 1 + 2 + 2) - (100 + 50) = -143$

Algorithm - Benefits

- easily parallelizable algorithm
- epoch number could be made as per resources or by some metacomputing
- will definitely find a better solution than normal travelling, solution will become better and better with time.

Modelling Inter-City Road Network

We have also developed a way to model intercity road network for a random country. We designed it because using it one might think for good heuristics for the problem

Modelling Inter-City Road Network

First generate a random tree of N nodes using Prüfer Sequence. It will ensure that there exists at least one route between any two locations and Tree representation denotes nearby cities.

Modelling Inter-City Road Network

Generate weights (distance) for edges generated in the previous step, we assign weights to edges using a normal distribution as we observed that in the real world most of the distance between neighbouring cities is almost near an average value but some are very near and some are very far away.

$$D_1 \sim \mathcal{N}(\mu_1, \sigma_1^2) \tag{23}$$

$$D_1 > 0 \tag{24}$$

$$\mid D_1 - \mu_1 \mid \leq 3\sigma_1 \tag{25}$$

Modelling Inter-City Road Network

Divide all cities in our model in 3 categories, metros, non-metros and remote locations, we connect all metro cities with each other adding new edges, we do not add new edges for non-metro cities, and we also add new edges from metros to non-metros, All these cities are randomly selected. All the weights of new edges as per the following equation

$$D_2 \sim U((1-\alpha)minDis(A,B),\ (1+\alpha)maxDis(A,B)) \tag{26}$$

Modelling Inter-City Road Network

Lastly we need to assign a sense of time to each edge, we define time by

$$time = \frac{distance}{c \times max(degree(A), degree(B)) + S} \tag{27}$$

$$S \sim \mathcal{N}(\mu_2, \sigma_2^2) \tag{28}$$

$$S > 0 \tag{29}$$

$$| S - \mu_2 | \leq 3\sigma_2 \tag{30}$$

basic motivation is usually highly connected cities have better road infrastructure and we are using a normal distribution (about 60 Km/Hour) to simulate the real world.

How to make Billion Dollars? (;

Steps to become a billionaire

- Improve the algorithm and make it much more real-timen or develop a good appoximation algorithm
- Deploy this algorithm initially as B2B for goods
- Take some funding to open pickup centres in major cities
- Expand .. Expand .. Expand

Conclusion

We were able to develop an algorithm to solve the problem and we were able to give a mathematical framework of how cities are organized.
Our Algorithm is able to solve the problem for a small number of nodes and parcels,Hence, we would be able deploy this in between major cities of the country with effective results.

Further Challenges

- To design heuristics and metaheuristic based algorithms/approximation-algorithms to solve this problem quickly
- To try other approaches than search formulation
- To try other variants of this problem

Acknowledgements

- Dr. Arti Pandey (Assistant Professor, Department of Mathematics, IIT Ropar), would not have be able to come up with this idea if not worked her on CarPooling Problem in Summers of 2019.
- Rajat Sharma (B18CSE039), for review and valuable suggestions.


Thank
you