Travelling salesman problem considering the weight of vertices

Kristie Harris
Rutgers University
Piscataway, NJ, USA
Email: kf.harris@rutgers.edu

Xuenan Wang
Rutgers University
Piscataway, NJ, USA
Email: xuenan.roderick.wang@rutgers.edu

Zhenyuan Zhang
Rutgers University
Piscataway, NJ, USA
Email: zzy.zhang@rutgers.edu

***Abstract*— In our daily life, we always try to find the optimal route to save time and energy. That is why GPS navigator is necessary when travelling. There is so-called classical travelling salesman problem (TSP) discussing how to find the optimal route that include all vertices once and finally goes back to the starting point. Sometimes, however, not only do we consider the distance cost, we might even care more about the energy cost coming from the mass that is carried. For example, a UPS driver needs to consider the total mass of packages on the truck as well as the distance of different routes. That is why in this paper we try to modify the TSP algorithm so that it can be applied on a model in which not only edges but also vertices themselves are weighted.

I. Project Description

In this paper, we modify TSM algorithms for real life applications. The model used here is undirected, connected, weighted graph. All the edges are weighted positively(representing distance). Also vertices are weighted as well. The weight at each vertex could be positive or negative depending on different situations. In garbage collection, for example, the weight will be the mass of garbage collected at each location and is positve. For delivery men, on the other hand, the weight will then be the mass of packages delivered to each location and is negative. Our purpose is to find the optimal route to visit all vertices considering the weight of both vertices and edges. The main difference from original weighted graph is that, the weight of edges only adds into the cost when edges are visited. But the weight of vertices defined in this project will continuously raise the total cost during the rest of tour. This project can be used in real life to save the cost of garbage collection, package delivery, etc. This is a NP-Hard problem and it's not easy for un-highly-educated man to solve and that's the value of out project, which is help to solve this complicate problem by simply need user provide basic information like how far from A to B and what's the weight of P1. The outcome of this project will not need any basic of computer and simple typing skill will do.

The project has four stages: Gathering, Design, Infrastructure Implementation, and User Interface.

A. *Stage1 - The Requirement Gathering Stage.*

- The general system description: Package delivery has always been a tricky task. Delivery men's job is to deliver packages as soon as possible. At the meantime, their employers expect them to use as few resources like gasline and vehicle abrasion as possible so that they can actually make more money. This is very similar to the traveling salesman problem (TSP). The TSP originally ask the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?" Isn't that similar with the package delivery problem as we just mentioned? The difference of these two problems is that each city in TSP doesn't actually affect later cost, but in package delivery problem, it does. To understand the difference, we need to figure out what exactly is this package delivery problem we have been talking about. Assume a delivery man leaves from package warehouse and has 20 packages (P1 P20) with him. He needs to go to 7 places (A G) to deliver these packages. There are some roads (R1 Rn) he can choose and those roads are not the same length. Considering the delivery man needs to deliver these packages as soon as possible, what is his best way to chose the route? Or even more, if we consider the extra package weight will cause vehicle consuming more gas, and delivery man needs to make sure he can deliver packages fast and economical as well. What route is best for this scenario?
- The real world scenarios:
  - Scenario1 description: Package delivery arrangement and optimization.
  - System Data Input for Scenario1: Package weight (P1 P20), Road length (R1 Rn)
  - Input Data Types for Scenario1: Matrix
  - System Data Output for Scenario1: A functional road selection collection
  - Output Data Types for Scenario1: Array
  - Scenario2 description: Garbage collection arrangement and optimization.
  - System Data Input for Scenario2: Garbage weight (P1 P20), Road length (R1 Rn)
  - Input Data Types for Scenario2: Matrix
  - System Data Output for Scenario2: A functional road selection collection
  - Output Data Types for Scenario2: Array
- Project Time line and Divison of Labor. This project will be divided into 3 parts. First one is to try to build a

working map to simulate the delivery progress. Secondly, we need to try to figure out what is the best strategy to deliver packages without considering gas consumption. Last part, we take package weight into consideration and try to analyze data to come up with a functional resolution. Each part will take about one week to finish. For now, we are thinking Xuenan and Zhenyuan will be in charge of coding and algorithm design, Kristie will be in charge of report writing and also algorithm improvements. Both three of us will be participating in presentation preparation.

B. *Stage2 - The Design Stage.*

Transform the project requirements into a system flow diagram, specifyng the different algorithms, data types and structures required for processing and their associated operations. The deliverables for this stage include the system flow diagram containing a graphical representation and textual descriptions of the corresponding data trasnformations, high level pseudo code of the overall system operation, and overall system time and space complexity.

Please insert your deliverables for Stage2 as follows:

- Short Textual Project Description. Please insert here the flow diagram textual description here together with its overall time and space complexity.
- Flow Diagram. Please insert your system Flow Diagram here.
- High Level Pseudo Code System Description.
  Initially, a complete directed graph is given. It has weight on edges and vertices.(distance cost and mass cost)

  1. For small n, we can use the easiest method which is exhaustion:

```
#Set the starting point
#Generate all permutations of visiting n vertices
#Keep track of the minimum cost considering weight
of both edges and vertices
#Return the minimum cost and its path
```

  This method has time and space complexity of O(n!) since there are n! permutations in total(That's why it only works for small n). This method will be used to verify our result using the other two methods.

  2. A more realistic and effective way is dynamic programming:

```
#Set the starting point
#Start with visiting v=2 vertices in total.(1 starting
point and 1 other point)
#Calculate the cost of each path and store the
result.(All vertices visited already, the last vertex
visited, total distance cost, total mass cost, mass
carried)
#If more than one path share the same visited
vertices and last vertex, only keep the one with
minimum cost
#Continue with v=3 using the result of v=2
#Repeat the previous steps for v=4, 5, 6... until v=n
#The last step is adding the edge from the end point
back to the starting point and find the minimum cost
#Return the minimum cost and its path
```

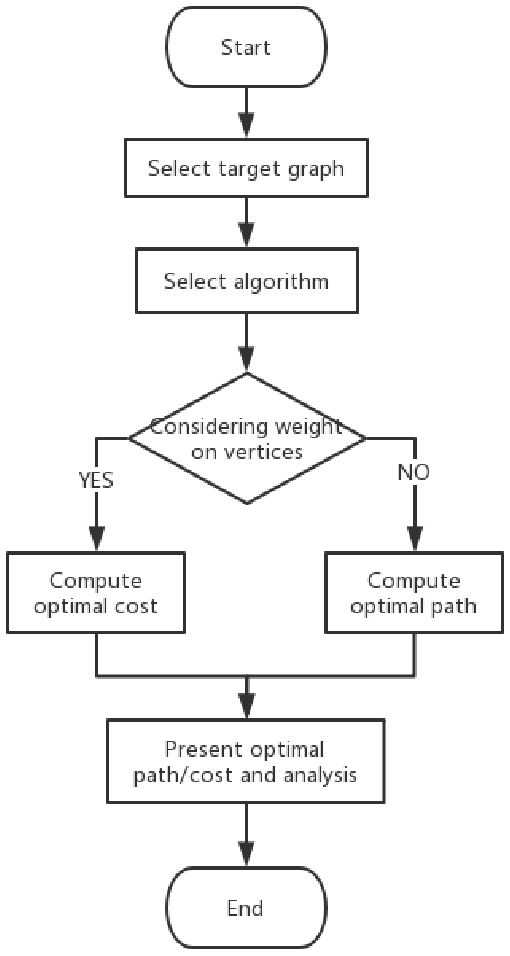

Fig. 1. System Flow Diagram

This method is much faster than the first one and it should give the correct final answer. But still it is not suitable for n that is too large.

3. An even faster method is greedy algorithm:

```
#Set the starting point and let end point to be the
same point
#Among all the edges coming out of the end point,
find the one with minimum total cost. Include that
```

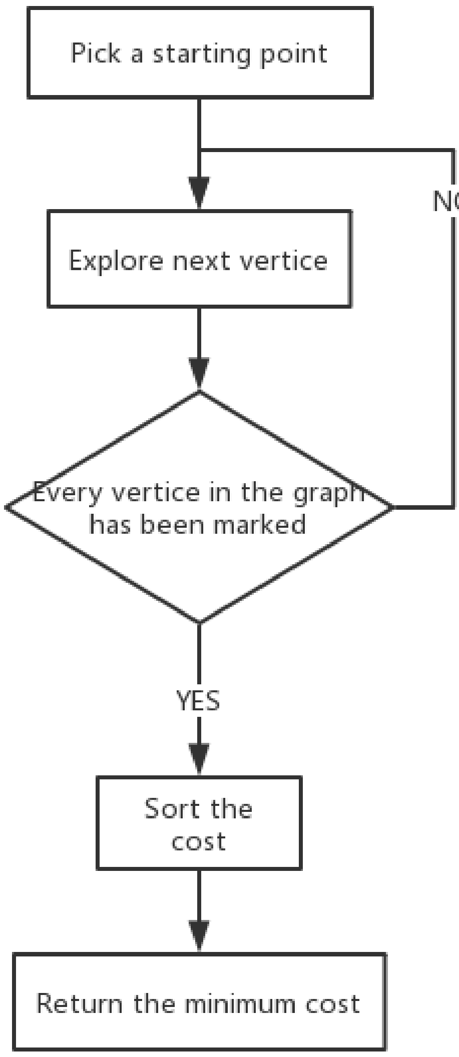

Fig. 2. Algorithm 1: Exhaustion

edge into the path and set the other node of the edge to be new end point
#Repeat the last step with the new end point until all vertices are visited
#Return the cost and the path

This method does not necessarily return the global minimum cost. But with some assumptions, it is still a good approximation and it is really fast!

- Algorithms and Data Structures.
1. Brute-force approach: explore every possible permutation to find the best solution. Easy but time-consuming.

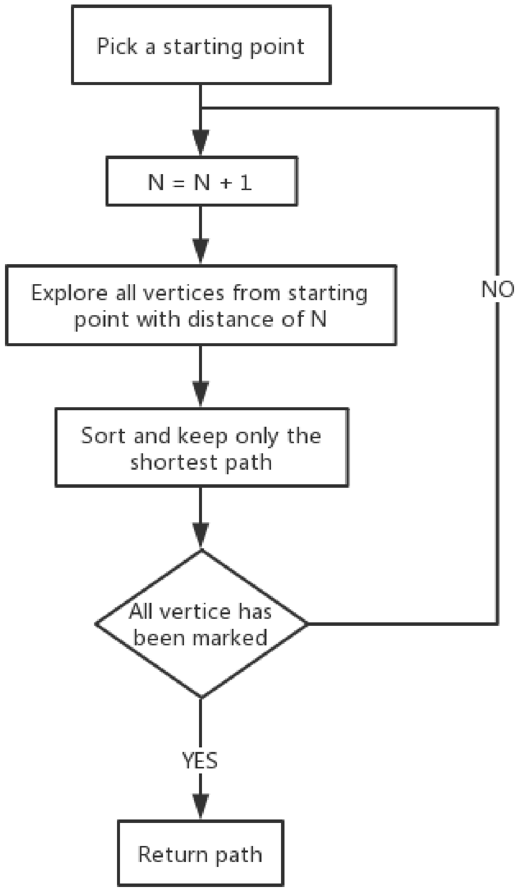
Pick a starting point
N = N + 1
Explore all vertices from starting point with distance of N
Sort and keep only the shortest path
NO
All vertice has been marked
YES
Return path

Fig. 3. Algorithm 2: Dynamic Programming

Data Structures: Graph(Adjacency Matrix), Tree, Array

2. Dynamic programming: Divide the problem into subproblems. Start with 1 point in the path and continue adding more points until all vertices are visited. Exponential time complexity.
Data Structures: Graph(Adjacency Matrix), Cost Matrix, Binary integer, Array

3. Approximate algorithm: Greedy algorithm always find local minimum cost. It might not give global best solution. But it is fast.
Data Structures: Graph(Adjacency Matrix), Tree, Array

- Flow Diagram Major Constraints. Please insert here the integrity constraints:
  - Input Constraint. Input graph must be fully connected directed graph.
  - Distance Constraint. Distance must be positive integer.

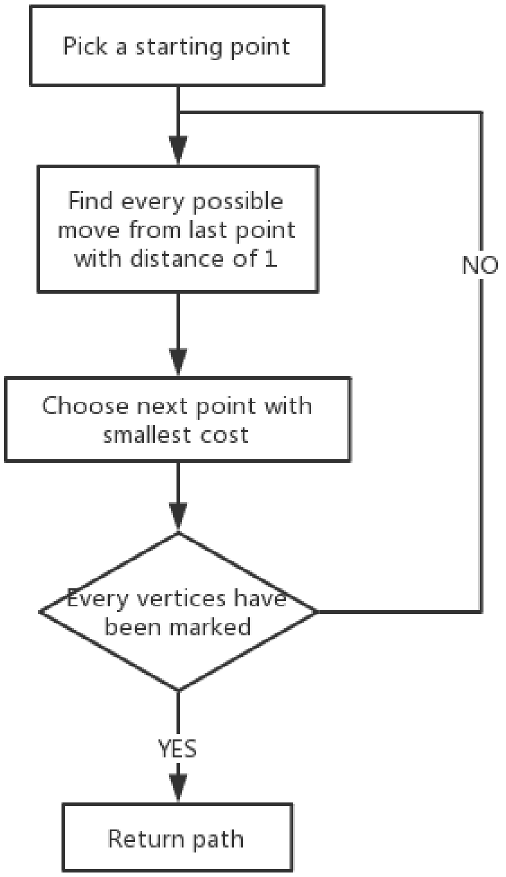
Pick a starting point
Find every possible move from last point with distance of 1
Choose next point with smallest cost
Every vertices have been marked
NO
YES
Return path

Fig. 4. Algorithm 3: Greedy Algorithm

- Weight Constraint. Weight must be positive integer.
- Output Constraint. Output must be an array of path.

C. *Stage3 - The Implementation Stage.*

Specify the language and programming environemnt you used for your implementation. The deliverables for this stage include the following items:

- Sample small data snippet.
- Sample small output
- Working code
- Demo and sample findings
  - Data size: In terms of RAM size; Disk Resident?; Streaming ?;
  - List the most interestng findings in the data if it is a Data Exploration Project. For other project types consult with your project supervisor what the corresponding outcomes shall be. Concentrate on demonstrating the Usefuness and Novelty of your application.

D. *Stage4 - User Interface.*

Describe a User Interface (UI) to your application along with the related information that will be shown on each interface view (How users will query or navigate the data and view the query or navigation results). The emphasis should be placed on the process a user needs to follow in order to meet a particular information need in a user-friendly manner. The deliverables for this stage include the following items :

- The modes of user interaction with the data (text queries, mouse hovering, and/or mouse clicks ?).
- The error messages that will pop-up when users access and/or updates are denied
- The information messages or results that wil pop-up in response to user interface events.
- The error messages in response to data range constraints violations.
- The interface mechanisms that activate different views in order to facilitate data accesses, according to users' needs.
- Each view created must be justified. Any triggers built upon those views should be explained and justified as well. At least one project view should be created with a justification for its use.

Please insert your deliverables for Stage4 as follows:

- The initial statement to activate your application with the corresponding initial UI screenshot
- Two different sample navigation user paths through the data exemplifying the different modes of interaction and the corresponding screenshots.
- The error messages popping-up when users access and/or updates are denied (along with explanations and examples):
  - The error message:
  - The error message explanation (upon which violation it takes place): Please insert the error message explanation in here.
  - The error message example according to user(s) scenario(s): Please insert the error message example in here.
- The information messages or results that pop-up in response to user interface events.
  - The information message: Please insert the error message in here.
  - The information message explanation and the corresponding event trigger
  - The error message example in response to data range constraints and the coresponding user's scenario Please insert the error message example in here.
- The interface mechanisms that activate different views.
  - The interface mechanism: Please insert the interface mechanism here.

II. PROJECT HIGHLIGHTS.

- Only working applications will be acceptable at project completion. A running demo shoul be presented to your project advisor at a date to be specified after the second midterm. A version of your application shall be installed in a machine to be specifed later during the semester. Your final submissiom package will also include a final LaTeX report modeled after this document, as well as a Power Point Presentation.
- The presentation (7 to 8 minutes) should include at least the following items (The order of the slides is important):
  1) Title: Project Names (authors and affiliations)
  2) Project Goal
  3) Outline of the presentation
  4) Description
  5) Pictures are essential. Please include Interface snapshots exemplyfing tthe different modes of users's interaction.
  6) Project Stumbling Blocks
  7) Data collection, Flow Diagram, Integrity Constraints
  8) Sample Findings
  9) Future Extensions
  10) Acknowledgements
  11) References and Resources used(libraries, languages, web resources)
  12) Demo(3 minutes)

  Please follow the sample presentation mock up that is posted on Sakai.
- By Dec 1 your group should have completed the final submission. This includes a presentation (7 to 8 minutes) to your project advisor as well as a convincing demo of your project functionalities (3 minutes): every group member should attend the demo (and presentation) indicating clearly and specifically his/her contribution to the project. This wil allow us to evaluate all students in a consistent and fair manner.
- Thank you, and best of luck!