INDIAN STATISTICAL INSTITUTE

M. TECH. THESIS

A Transportation Scheduling Algorithm

Author:
Arghya Bhattacharjee

Supervisor:
Dr. Sandip Das

A thesis submitted in fulfillment of the requirements for the degree of Master of Technology

in the

Computer Science Department
Indian Statistical Institute

July 19, 2016

Declaration of Authorship

I, Arghya Bhattacharjee, declare that this thesis titled, “A Transportation Scheduling Algorithm” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an M. Tech. degree in Computer Science at this institute.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this institute or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Abstract

Suppose there are pick-ups and drops along a line. Each pick-up has weight to be transported to its corresponding drop. A carriage with a finite capacity is assigned to carry out this transportation job. Our objective is to find out the traversal route for the carriage for which the distance traversed by the carriage is minimum. Here we have tried to find out algorithms which can serve our objective.

Acknowledgements

I take this opportunity to thank Prof. Sandip Das (ACMU Unit, ISI Kolkata) for his valuable guidance and inspiration. As a mentor, his work is exceptional. He suggested me to work on this problem and motivated me throughout the process. Finding the right problem is really a tough task in research and he has done this favour for me. He introduced this problem to me and showed me the right path wherever I got stuck. He gave me his valuable time and provided me basic structure to the problem on which I became able to add more results. I am grateful to him for giving me chance to work under his supervision and will feel myself lucky if I could work with him in future.

I would like to thank all my faculties for providing me enough time to think on this problem

Finally I would like to thank all my batch mates for their amazing company.

Contents

To you…

Chapter 1

Introduction

Suppose there are pick-ups and drops along a line. Each pick-up has weight to be transported to its corresponding drop. A carriage with a finite capacity is assigned to carry out this transportation job. Our objective is to find out the traversal route for the carriage for which the distance traversed by the carriage is minimum. Another way to state our objective is that, we want to minimize the fuel cost of the carriage. Here we have tried to find out algorithms which can serve our objective. In this chapter, we will define one very basic and special scenario of this problem, and discuss about what other generalizations can be done over that scenario.

1.1 Problem Definition

Suppose we place $2n$ number of points on a line, where n is any natural number. We mark any n of those $2n$ points as pick-ups and the rest n points as drops. Now, from left to right, we mark the pick-ups as p_i, and the drops as d_i, where i varies from 1 to n in each case. The only constraint we follow until now is that, for every value of i, d_i lies somewhere to the right of p_i on the line. Now, suppose, for every value of i, there is a unit weight placed at p_i, which is to be transported to d_i. The carriage, which is assigned to do this transportation job, is also of unit capacity. It will start from the left most of the $2n$ points, do the whole transportation job, and then return to the starting point again. Our objective is to come up with an algorithm, which will determine the traversal route for the carriage of minimum length.

1.2 Some Possible Generalizations

Here are some possible generalizations we can think about, over the scenario we just defined.

1. The carriage capacity can be more than one.

2. For any value of i, d_i may be placed somewhere to the left of p_i as well.

3. Any pick-up can have any number of unit weights.

4. All the unit weights from one pick-up may not have to be delivered to a single drop, i.e., there can be more than one drops corresponding

to a single pick-up.

5. Unit weights from more than one pick-ups may have to be delivered to a single drop, i.e., there can be one drop corresponding to more than one pick-ups.

6. The carriage may have to start from any of the $2n$ points, and end up at any one of them as well.

1.3 Some Useful Definitions

Here are some definitions which we have used throughout our subsequent discussions.

1.3.1 Segment

A segment is a part of the line in between two consecutive points.

1.3.2 Bridge

A bridge is a segment such that there is no pick-up anywhere to the left of that segment whose corresponding drop is somewhere to the right of that segment.

1.3.3 Island

An island is any maximum possible stretch of points containing no bridge.

In the next chapter, we will propose an algorithm for the basic scenario and prove its correctness.

Chapter 2

Algorithm for the Basic Scenario

Here is the algorithm proposed for the basic scenario we discussed about in the last chapter.

2.1 An Observation

The left most point of any island cannot be a drop, because its corresponding pick-up must lie somewhere to the left of that drop, but inside the same island. Similarly, the right most point of any island cannot be a pick-up, because its corresponding drop must lie somewhere to the right of that pick-up, but inside the same island. So, it is clear that, the left most point of any island is a pick-up and the right most point of any island is a drop.

2.2 The Algorithm

The carriage follows the following instructions throughout its traversal.

1. Whenever the carriage encounters an unserved pick-up, it picks up its weight, goes to its corresponding drop and drops the weight there.
2. Whenever the carriage drops an weight, it checks if the segment attached to that drop to its right is a bridge or not. If it is a bridge, the carriage starts moving towards right. Otherwise, the carriage starts moving towards left.

The carriage starts by picking up the weight from the left most pick-up, i.e., the point it starts from, goes to its corresponding drop, and drops the weight there.

2.3 Proof of the Algorithm

We will prove the algorithm in three parts. First, we will prove that the algorithm will terminate after a finite time. Then we will prove that, once the algorithm terminates, all the points are served. Finally we will prove that the carriage traverses the path of minimum possible length. But first of all we will go through another observation before going into the detail of proof.

2.3.1 Another Observation

Suppose at some point of time, the carriage has come to a point on the line. Let us call that point x. And suppose there is a point y which lies somewhere to the left of x on the line. Now, if we know that the algorithm has terminated, i.e., the carriage has returned to its starting point, and we also know that at some point of time, the carriage was at the point x, then we can say that the point y must has been served by the carriage. Because, if y is a pick-up, the carriage must have gone through it during a backward journey. If y was unserved until that time, the carriage must have served it then. Similarly, if y is a drop, then the carriage must have gone through its corresponding pick-up during a backward journey, because that pick-up lies somewhere to the left of y. If that pick-up was unserved until that time, the carriage must have served it then. So, in any case, y must have been served by the carriage.

2.3.2 Proof of termination

The carriage moves towards right only in two cases. Firstly, if it encounters an unserved pick-up, and secondly, if it drops weight in a drop which has a bridge attached to it to its right. In the first case, the journey towards right terminates when the corresponding drop is reached, and in the second case, the journey towards right terminates just after crossing the bridge. So, in each of the cases, the journey towards right terminates after a finite time. As both the number of pick-ups and the number of drops are finite, the overall journey towards right is also finite. As it moves towards left otherwise, after a finite time, it returns to the left most point.

2.3.3 Proof of the fact that once terminated, all the points are served

Suppose the algorithm has terminated and there is a pick-up which is still unserved. Let that pick-up be p_j. Let us consider the right most drop of that island, which has been served. Let that drop be d_k. We already know that, once the carriage reaches a point on the line, it will certainly serve all the pick-ups to the left of it. Now, if d_k is the right most drop of its island, then p_j must have been served. Else, let us consider all the drops which are in the same island of d_k but to the right of d_k. Now, if all of their corresponding pick-ups are towards right of d_k as well, then there is a bridge attached to d_k to its right, which contradicts the definition of an island. Else, there is at least one drop in the same island of d_k and to the right of d_k, whose corresponding pick-up is to the left of d_k. Then, d_k cannot be the right most drop of that island which is served. So, d_k must be the right most drop of that island, and as a result, all the pick-ups of that island are served. Also, as the carriage goes to the corresponding drop to drop the weight immediately after picking up an weight from any pick-up, all the drops of that island are also served. Now, as the argument is valid for any island on the line, it proves that, once the algorithm terminates, all the points on the line are served.

2.3.4 Proof of traversing the path of minimum possible length

Here are two simple observations.

1. Each bridge is traversed only once towards right. Because, a bridge is traversed towards right only when the drop attached to that bridge to its left is served. As one drop is served exactly once, a bridge is traversed exactly once towards right.

2. Whenever each non-bridge segment is traversed towards right, the carriage is carrying weight. This is clearly ensured from our algorithm.

Now, we can see that, each bridge is traversed minimum number of times, i.e., once, towards right. Also, each non-bridge segment is traversed minimum number of times towards right. Because, if a non-bridge segment is traversed more than minimum number of times towards right, the carriage must cross the segment towards right with no weight at least once. But our algorithm ensures that, whenever each non-bridge segment is traversed towards right, the carriage is carrying weight. So, the carriage traverses each segment minimum number of times, which ensures that the carriage overall traverses the path of minimum possible length.

In the next chapter, we will discuss about one possible generalization of the already discussed scenario.

Chapter 3

A Possible Generalization

Now we will discuss about the first possible generalization that we mentioned in chapter 1, i.e., the carriage has capacity which is not necessarily 1, but anything greater or equal to 1. Let us call our new carriage capacity c where c is any natural number. This time we will describe the setup in a slightly different way first. Then we will proceed to try to achieve our objective.

3.1 The Setup

This time we add two more segments to the line. One is attached to the left of the left most point and the other is attached to the right of the right most point. We define a variable y for each segment as the minimum number of times the carriage must cross that segment towards right in order to carry out the whole transportation job. Suppose there are x pick-ups to the left of any segment, such that, each of their corresponding drops is somewhere to the right of that segment. Then the value of y of that segment is equal to the ceiling value of $\frac{x}{c}$. Let us call the value of y of the segment attached to any point to its left as y_{left} of that point and the value of y of the segment attached to any point to its right as y_{right} of that point. Now, if for any point, y_{right} is greater than y_{left}, we put an up arrow over that point. Similarly, if for any point, y_{right} is less than y_{left}, we put a down arrow over that point. As of now, we will limit our discussion only within the case where there is no bridge on the line. Later we will introduce bridges on the line.

3.2 The Algorithm

The carriage performs two kinds of journey, forward (towards right) and backward (towards left). During a forward journey, it serves each and every point it encounters, if possible, including the starting and finishing points of that forward journey. During a backward journey, it doesn't serve any point it encounters. The carriage starts from the left most point towards right to start the first forward journey. Each forward journey ends, and a new backward journey starts, when the carriage serves a drop which already has a down arrow. Each backward journey ends, and a new forward journey starts, when the carriage encounters an unserved pick-up which already has an up arrow. Whenever the carriage reaches a point during a forward journey, at the time of leaving that point, it decreases the y_{left} of that point by 1, and accordingly updates the arrow of that point, if required.
Let us go through some lemmas first, before discussing about the proof of the algorithm.

3.3 Lemmas

Lemma 3.3.1. *An up arrow is always on a pick-up.*

Proof. This is simply because of the fact that the value of y cannot increase after a drop. □

Lemma 3.3.2. *A down arrow is always on a drop.*

Proof. This is simply because of the fact that the value of y cannot decrease after a pick-up. □

Lemma 3.3.3. *The number of up arrows is equal to the number of down arrows.*

Proof. The value of y of the left most segment is 0. Similarly, the value of y of the right most segment is 0. Now, at each up arrow, the value of y increases by 1 and at each down arrow, the value of y decreases by 1. The value of y does not change at any other point. As the left most and the right most values of y are equal, the total number of up arrows must be equal to the total number of down arrows. □

Lemma 3.3.4. *Once a point has no arrow, it will never have any arrow.*

Proof. Suppose at some point of time, a point has no arrow over it. So, there will never be any change of direction of the carriage from that point. So, whenever the carriage will encounter that point, it will traverse both the segments attached to the point, one immediately after the other. So, whenever the value of y_{left} will decrease, the value of y_{right} will also decrease immediately after that. As a result, an up arrow will be generated over that point, only to get cancelled out immediately after generation. So, effectively no arrow will be generated over that point ever again. □

Lemma 3.3.5. *The carriage cannot reach the right most point with weight of any other drop.*

Proof. Suppose the carriage reaches the right most point with at least one weight of any other drop. Now, the drop of that weight is somewhere in between its pick-up and the last point, and as the carriage has been at its pick-up some time before reaching the last point, it must have already encountered its drop in a forward journey at least once, with the weight in the carriage. Then, the weight must have already been dropped. So, the carriage cannot reach the right most point with weight of any other drop. □

3.4 Proof of the Algorithm

Suppose the carriage traverses in such a way such that the number of times each segment is crossed in the forward direction is equal to the value of y of that segment. Then we say that theoretically minimum amount of path has been traversed. Now, a drop can be served in two ways. First, the carriage reaches the drop during a forward journey, serves it, and then continues its forward journey. Second, the carriage reaches the drop during a forward journey, serves it, and then starts the next backward journey. Similarly, a pick-up can be served in two ways. First, the carriage reaches the pick-up during a forward journey, serves it, and then continues its forward journey. Second, the carriage reaches the pick-up during a backward journey, serves it, and then starts the next forward journey. Now, suppose there is a down arrow on a drop. Then, whenever the drop will be reached during a forward journey and served, the next backward journey will start. So, if a drop has a down arrow, it cannot be served in the first way. But, if a pick-up has an up arrow, it can be served in both the ways. Now, if a pick-up has an up arrow, and is served in the first way, then the situation is beyond the scope of our algorithm. Such an example is $p_1p_2p_3p_4d_1d_3p_5d_2d_4d_5(c = 2)$. Now, the carriage can only reach the right most point in two ways. First, with only the weight of the last point. Second, with no weight. Now, if the carriage reaches the right most point with no weight, then the situation is beyond the scope of our algorithm. Two such examples are $p_1p_2p_3p_4d_1d_3d_2d_4(c = 2)$ and $p_1p_2p_3p_4d_1d_2p_5p_6d_3d_5d_4d_6(c = 2)$. In case of the first example, the carriage never changes its direction. And in case of the second example, the carriage changes its direction before reaching the right most point with no weight. Now, suppose the algorithm does not terminate. Then it can do so in two ways. First, with no change in direction. Second, with at least one change in direction. The first case is not possible. Because, it implies that the carriage ultimately reaches the last point with its weight, which will cause a change in direction. Similarly, the second case is also not possible. So, the algorithm will terminate. Now, suppose the algorithm has terminated, and at least one pick-up is still unserved. If this happens, then the situation is beyond the scope of our algorithm. Such an example is $p_1p_2p_3p_4d_4d_1d_2d_3(c = 2)$. Now, suppose the algorithm has terminated, all the pick-ups have been served, but at least one drop is still unserved. Then, the carriage will return to the left most point with at least one weight. If this happens, then the situation is beyond the scope of our algorithm. Such an example is $p_1p_2p_3p_4p_5d_1d_2d_3d_4d_5(c = 2)$. We assume that we never face any of the above mentioned situations which are beyond the scope of our algorithm. In that case, the algorithm will terminate, and once it terminates, all the points will be served. Now, suppose the algorithm has terminated. Then all the arrows are gone. Then y of each segment is equal. As y of the left most segment is 0, then y of each segment is 0. So, the carriage traverses theoretically minimum amount of path.

3.5 Discussion

1. It can easily be observed that, in each of the examples given here, it's impossible for the carriage to serve all the points traversing theoretically minimum amount of path. But such examples can also be constructed where it is possible for the carriage to serve all the points traversing theoretically minimum amount of path, but not by following our algorithm. Such an example is $p_1p_2p_3p_4p_5d_3d_1d_2d_4d_5(c = 2)$. So, we can conclude the following about the working of our algorithm:

   (a) Whenever it's impossible for the carriage to serve all the points traversing theoretically minimum amount of path, our algorithm will not work.

   (b) Whenever our algorithm will work, it's possible for the carriage to serve all the points traversing theoretically minimum amount of path.

   (c) Whenever it's possible for the carriage to serve all the points traversing theoretically minimum amount of path, there is no guarantee that our algorithm will work.

2. Another observation is that, after starting from the left most point, if the carriage changes direction from all the arrows, then the algorithm terminates and the carriage serves all the points traversing theoretically minimum amount of path, whatever the order is. But sometimes, it's impossible to do so whatever order we use, and sometimes, our algorithm doesn't follow the correct order, like in case of the last example given.

3. We have already seen that the carriage cannot reach the right most point with weight of any other drop. So, in any case, the carriage will be empty after reaching there and serving it, if possible. So, if there are bridges in the line, the carriage comes to the right most point of one island, serves it if possible, crosses the bridge, serves the next island and then comes back to the right most point of the previous island to do the remaining transportation work. This part is similar to the algorithm discussed in chapter 2.

4. In any of the algorithms discussed so far, there has never been any case where we have to consider the length of any segment. This is because of the fact that the lengths of the segments have no effect in determining the path with minimum possible length.