1988

A Note on the Complexity of a Simple Transportation Problem

Greg N. Frederickson
Purdue University, gnf@cs.purdue.edu

Report Number:
88-809

A NOTE ON THE COMPLEXITY OF A SIMPLE TRANSPORTATION PROBLEM

Greg N. Frederickson

CSD-TR-809
September 1988
(Revised March 1990)

A NOTE ON THE COMPLEXITY OF A SIMPLE TRANSPORTATION PROBLEM*

Greg N. Frederickson

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

* This research was supported in part by the National Science Foundation under Grant CCR-8620271, and by the Office of Naval Research under contract N00014-86-K-0689.

Abstract. Consider the problem of using a vehicle to transport k objects one at a time between s stations on a circular track. Let the cost of the transportation be the total distance traveled by the vehicle on the track. An $O(k + M(s, q))$ time algorithm is presented to find a minimum cost transportation, where $M(m, n)$ is the time to solve a minimum spanning tree problem on a graph with m edges and n vertices, and $q \leq \min\{k, s\}$ is the number of strongly connected components in an associated balanced problem. Also, the minimum spanning tree problem on a graph with m edges and n vertices is reduced to a transportation problem on a linear track with $O(m)$ stations, $O(m)$ objects and $O(n)$ strongly connected components in $O(m)$ time.

Key words and phrases. transportation problems, robot arm motion, circular track, graph augmentation.

1. Introduction

Consider an undirected weighted graph with objects located at various vertices. Associated with each object is a destination vertex, to which that object is to be moved by a vehicle that traverses the edges of the graph. A fundamental problem in motion planning is to determine a minimum cost tour of the vehicle that transports all objects from their initial positions to their destinations. In the case of general graphs, the problem is NP-hard, even if the vehicle can transport only one object at a time [FHK]. Recently, attention has focused on solving this motion planning problem on very simple classes of graphs, with unit capacity vehicles. Such examples have potential applications in robotics. Atallah and Kosaraju consider graphs that are simple paths and simple cycles [AK]. Frederickson and Guan consider graphs that are trees [FG]. Both papers distinguish between two cases, based on whether or not drops are allowed in the transportation. A *drop* is an unloading of an object at a vertex that is not its destination. If an object is dropped, its move is not immediately completed, and the object must be picked up and transported farther at some later time in the transportation.

In this note we tighten the bound of [AK] for the case of simple cycles with no drops. Whereas all possible solutions are considered in a divide-and-conquer search that is used in [AK], we quickly prune away all but a constant number of possible solutions. We also show that the asymptotic complexity of the transportation problem with no drops for either a simple path or a simple cycle is essentially the same as that of the minimum spanning tree problem (on a general graph).

For simplicity, we shall use much of the notation in [AK]. In particular, we

shall refer to the underlying graph as a track, and the vertices in the graph as stations. Let k be the number of objects, s the number of stations, and q the number of strongly connected components once additional moves are added to yield a certain "balanced" problem. Let $M(m, n)$ be the time to solve the minimum spanning tree problem on a graph with n vertices and m edges. (Currently, the fastest known algorithm for the minimum spanning tree problem is given in [GGST], providing an upper bound on $M(m, n)$ of $O(m \log \beta(m, n))$, where $\beta(\cdot, \cdot)$ is a slowly growing function similar to the $\log^*(\cdot)$ function.) For the case in which drops are allowed, an $O(k + s)$ time algorithm is given in [AK] for the circular track, and hence the linear track. For the case in which no drops are allowed, an $O(k + s \log s)$ time algorithm is given in [AK] for a circular track, and an $O(k + M(s, q))$ time algorithm for a linear track (where the time is as indicated in the note added in proof in [AK]).

We make several observations about the structure of the transportation problem on a circular track with no drops that allows us to generate an $O(k + M(s, q))$ time algorithm. We also provide a simple argument that solving a transportation problem on a linear track where $k = s$ is in general no easier than solving a minimum spanning tree problem on q vertices and s edges. Thus in the sense of asymptotic complexity, the circular track problem with no drops is no harder than the linear track problem with no drops. Thus for both the case of drops and the case of no drops, restricting the graph from a circular track to a linear track will make the problem no easier.

2. A faster algorithm

In this section we derive several observations that lead to a faster algorithm. We first recall some definitions from [AK]. Let the stations be indexed from 1 to s, and the edge between stations i and $i+1$ be denoted as interval $(i, i+1)$. Assume that each object moves in the shorter of the two directions for it, either clockwise or counter-clockwise, around the cycle. Let $\phi(i)$ be the *input flux* across interval $(i, i+1)$, defined as the number of clockwise moves in the input minus the number of counterclockwise moves in the input across interval $(i, i+1)$.

Recall from [AK] that in any transportation, the number of clockwise moves across an interval minus the number of counterclockwise moves across an interval will be the same for all intervals. More generally, for any set of moves in which the difference across every interval is the same, this common difference is called the *flux*. By adding moves in which the vehicle carries no object, any particular value of flux can be achieved. Let ψ be the value of the flux for some set of moves. Let l_i be length of interval $(i, i+1)$. Let $db(\psi)$ be the cost of a minimum cost set of augmenting moves that yield a flux of ψ. The function $db(\cdot)$ represents the cost to achieve degree balance between incoming and outgoing moves at each station. Then $db(\psi) = \sum_{i=1}^{n} |(\psi - \phi(i))| \; l_i$. Note that adding moves with the "empty object" to achieve degree balance may result in more than one strongly connected component, where each component is Eulerian, and each is isolated from the others. Among all minimum-cost sets of augmenting moves that yield a flux of ψ, let q_ψ be the minimum number of strongly connected components that result from any of these augmentations. There is a

minimum-cost set of augmenting moves that achieves flux ψ, creates q_ψ strongly connected components, and is of cardinality $O(k+s)$, and such a set can be found in $O(k+s)$ time. Besides the augmenting moves that achieve a particular flux, additional moves with no object are in general necessary to achieve connectivity among the components, and thus be able to construct a transportation.

We introduce some additional notation. Let c_ψ be the total length of intervals with $\phi(i) = \psi$. Then $c_\psi = \sum_{\phi(i)=\psi} l_i$. Note that $c_\psi \geq 0$ for all ψ. Let $tc(\psi)$ be the cost of a minimum cost set of augmenting moves that yield a transportation with flux ψ. The function $tc(\cdot)$ represents the cost to achieve both degree balance and connectivity. Let γ be the largest value of flux for which a set of augmenting moves of overall minimum cost achieves degree balance. We note some simple relationships among these quantities in the following lemmas.

Lemma 1. For all values ψ of flux, $tc(\psi) \leq db(\psi) + 2c_\psi$, which holds with strict inequality if $db(\psi) < tc(\psi)$.

Proof. The addition of the augmenting moves that achieve degree balance will leave one or more strongly connected components. If there is more than one such component, then the components are separated by intervals $(i, i+1)$ with $\phi(i) = \psi$. Adding a move in each direction across all but one such interval will yield one strongly connected component. □

Lemma 2. For all values ψ of flux, $db(\psi) = db(\psi - 1) - \sum_{j>\psi-1} c_j + \sum_{j\leq\psi-1} c_j$

Proof. Given a set of moves that achieve degree balance for flux $\psi - 1$, one can generate a set of moves that achieve degree balance for flux ψ by removing an augmenting move from each interval i for which $\phi(i) > \psi - 1$, and by adding an augmenting move for each interval i for which $\phi(i) \leq \psi - 1$. □

Lemma 3.

a. $\sum_{j>\gamma-1} c_j \geq \sum_{j\leq\gamma-1} c_j$

b. $\sum_{j\leq\gamma} c_j > \sum_{j>\gamma} c_j$

Proof. Part (a) follows directly from Lemma 2 and the definition of γ. Part (b) follows by combining the definition of γ with the following, which is obtained from Lemma 2.

$$db(\gamma+1) = db(\gamma) - \sum_{j>\gamma} c_j + \sum_{j\leq\gamma} c_j \quad . \square$$

Next we note that $db(\psi)$ is a concave (upwards) function of ψ.

Lemma 4. The function $db(\cdot)$ is concave.

Proof. A simple proof by induction on n establishes that functions of the form $\sum_{i=1}^{n} |a_i x - b_i|$ are concave. Note that $db(\cdot)$ is is of this form. □

Let δ be the largest value of flux for which a minimum cost transportation is achieved. Then $tc(\delta) = \min_{\psi}\{tc(\psi)\}$. We next show that at most three values of ψ need be considered.

Lemma 5. A value of ψ in the range $\gamma - 1 \leq \psi \leq \gamma + 1$ achieves the minimum value of

$tc(\cdot)$.

Proof. Suppose $\delta \geq \gamma + 2$. Then

$$db(\delta) \leq tc(\delta - 1)$$

which by Lemma 1

$$\leq db(\delta - 1) + 2c_{\delta-1}$$

$$= db(\delta) + \sum_{j>\delta-1} c_j - \sum_{j\leq\delta-1} c_j + 2c_{\delta-1}$$

$$= db(\delta) + (\sum_{j>\gamma} c_j - \sum_{j\leq\gamma} c_j) - 2\sum_{j=\gamma+1}^{\delta-2} c_j$$

By the nonnegativity of c_ψ, $c_j \geq 0$ for $\gamma + 1 \leq j \leq \delta - 2$. But then $\sum_{j\leq\gamma} c_j - \sum_{j>\gamma} c_j \leq 0$, which contradicts Lemma 3b. Thus $\delta < \gamma + 2$.

Suppose $\delta \leq \gamma - 2$ and $db(\delta + 1) = tc(\delta + 1)$. Then by Lemma 4

$$db(\delta) \geq db(\delta + 1) = tc(\delta + 1)$$

Thus the cost of a solution with flux $\delta + 1$ is always at least as good as a solution with flux δ, a contradiction to the choice of δ.

Suppose $\delta \leq \gamma - 2$ and $db(\delta + 1) < tc(\delta + 1)$. Then

$$db(\delta) \leq tc(\delta + 1)$$

which by Lemma 1

$$< db(\delta + 1) + 2c_{\delta+1}$$

$$= db(\delta) - \sum_{j>\delta} c_j + \sum_{j\leq\delta} c_j + 2c_{\delta+1}$$

$$= db(\delta) + (\sum_{j\leq\gamma-1} c_j - \sum_{j>\gamma-1} c_j) - 2\sum_{j=\delta+2}^{\gamma-1} c_j$$

By the nonnegativity of c_ψ, $c_j \geq 0$ for $\delta + 2 \leq j \leq \gamma - 1$. But then $\sum_{j \leq \gamma-1} c_j - \sum_{j > \gamma-1} c_j > 0$, which contradicts Lemma 3a. From this and the preceding case, we may conclude that $\delta > \gamma - 2$.

Thus the above cases rule out all values ψ of flux except those in the range $\gamma - 1 \leq \psi \leq \gamma + 1$. □

The algorithm to solve the transportation problem is the following. First compute the values of $db(\psi)$ for values of ψ from $-k$ to k, as discussed in [AK]. Next perform a scan of the values $db(\psi)$ to identify γ. Then for each of the values $\gamma - 1$, γ, $\gamma + 1$ of flux, solve the associated minimum spanning tree algorithm, using the fastest currently known algorithm. Choose from among the three transportations the solution that is of smallest total cost. Let q be $\max\{q_{\gamma-1}, q_\gamma, q_{\gamma+1}\}$.

Theorem 1. The time to solve a transportation problem with no drops on a graph that is a simple cycle is $O(k + M(s, q))$, where k is the number of objects, s is the number of stations, q is the maximum of number of strongly connected components in three related balanced problems, and $M(m, n)$ is the time to solve the minimum spanning tree problem on a graph with n vertices and m edges.

Proof. By the discussion in [AK], the time to find γ is $O(k)$. The three minimum spanning tree problems can each be set up in $O(s)$ time. Since $M(s, \cdot)$ is $\Omega(s)$, the result follows. □

3. A reduction from the minimum spanning tree problem

We show how to reduce the minimum spanning tree problem to a transportation problem with no drops on a linear track. Recall from [AK] that the degree balanced version of the problem is one in which a minimum cost set of balancing moves has been added so that for any interval $(i, i+1)$ on the track, the number of moves across the interval in the clockwise direction equals the number of moves across the interval in the counterclockwise direction. Let k be the number of moves and s the number of stations. Among all such minimum-cost sets of augmenting moves, let q be the minimum number of strongly connected components that result from any of these augmentations. There is a minimum-cost set of augmenting moves that creates q strongly connected components and is of cardinality $O(k + s)$, and such a set can be found in $O(k + s)$ time.

Theorem 2. Let $R(k, s, q)$ be the time to solve a transportation problem with no drops on a linear track of s stations, with k moves, and q components in the balanced problem. The time to find a minimum spanning tree in a graph of m edges and n vertices is $O(R(m, m, n))$.

Proof. Let $G = (V, E)$ be a connected weighted undirected graph with m edges and $n > 3$ vertices. Without loss of generality, assume all edge weights are positive. Let W be the largest of the edge weights. Compute the degree of every vertex. For any vertex of degree less than 3, add edges of cost $W + 1$ to G to make every vertex be of degree at least 3. Once these edges have been added, if not all vertices are of even degree, introduce a new vertex, with edges of cost $W + 1$ to each vertex of odd degree. The

resulting graph G' will have n' vertices, $n \leq n' \leq n+1$, and m' edges, $m \leq m' \leq m + 3n$, and the degree of each vertex will be even and at least 4. By the choice of the cost of new edges, a minimum spanning tree of G' will be a minimum spanning tree of G, plus some edge to the new vertex if a new vertex was introduced. Given graph G', find an Euler tour of G', starting at any vertex. We denote the tour by the sequence of vertices and edges $v_0, e_1, v_1, e_2, v_2, \cdots, v_{m'}, e_{m'}, v_0$,

Given the Euler tour, we generate an instance P of a transportation problem on a linear track as follows. There will be $m' + 1$ stations, one for each visit of a vertex in the Euler tour. The edge from the j-th to the $(j+1)$-st station will correspond to the $(j+1)$-st edge in the Euler tour, and thus be of cost $c(e_{j+1})$. There will be one object originating at each station. The destinations of the objects are determined as follows. Consider the r-th station, and suppose it corresponds to a visit to vertex v in the Euler tour. Let the r'-th station correspond to the next visit to vertex v in the Euler tour. (If there is no next visit to v, let the r'-th station correspond to the first visit to v.) Then the destination of the object at station r is station r'. Since the degree of each vertex in G' is at least 4, every object will have a destination different from its originating station.

From the construction it is clear that for any vertex in G' the set of arcs in P form a cycle. It follows that there are n' strongly connected components in P. For each edge (v, w) in G' there is an edge of the same cost in the track from a station in the cycle of stations corresponding to v to a station in the cycle of stations corresponding to w.

Consider an optimal transportation Q for P. Consider the set of edges traversed by Q when no object is being carried. Any edge traversed in one direction is traversed in the other direction. The number of such edges is $n'-1$, and these edges correspond to a minimum spanning tree of G'. Thus we can solve a minimum spanning tree problem by generating an instance P, finding an optimal transportation Q for P, and extracting the edges of the minimum spanning tree from Q. Clearly, all steps other than that of finding the transportation will take $O(m')$ time. Thus the minimum spanning tree problem in G' can be solved in time $O(m' + R(m'+1, m'+1, n'))$. Assuming the monotonicity of $R(\cdot, \cdot, \cdot)$, and noting that $R(m, \cdot, \cdot)$ is $\Omega(m)$, this is $O(R(m, m, n))$. □

Acknowledgment. I would like to thank Mike Atallah for some helpful comments.

References

[AK] M. J. Atallah and S. R. Kosaraju, Efficient solutions to some transportation problems with applications to minimizing robot arm travel, *SIAM J. Computing 17* (1988) 849-869.

[FG] G. N. Frederickson and D.-J. Guan, Ensemble motion planning in trees, *Proc. 30th IEEE Conf. on Foundations of Computer Science*, Research Triangle, NC (1989) 66-71.

[FHK] G. N. Frederickson, M. S. Hecht and C. E. Kim, Approximation algorithms for some routing problems, *SIAM J. Computing 7* (1978) 178-193.

[GGST] H. N. Gabow, Z. Galil, T. Spencer and R. E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, *Combinatorica 6* (1986) 109-122.