To Fill or not to Fill: The Gas Station Problem. (Extended Abstract) *

Samir Khuller, Azarakhsh Malekian, and Julián Mestre

Department of Computer Science.
University of Maryland, College Park, MD 20742, USA.
Email: {samir,malekian,jmestre}@cs.umd.edu

Abstract. In this paper we study several routing problems that generalize shortest paths and the Traveling Salesman Problem. We consider a more general model that incorporates the actual cost in terms of gas prices. We have a vehicle with a given tank capacity. We assume that at each vertex gas may be purchased at a certain price. The objective is to find the cheapest route to go from s to t, or the cheapest tour visiting a given set of locations. Surprisingly, the problem of find the cheapest way to go from s to t can be solved in polynomial time and is not NP-complete. For most other versions however, the problem is NP-complete and we develop polynomial time approximation algorithms for these versions.

1 Introduction

Optimization problems related to computing the shortest (or cheapest) tour visiting a set of locations, or that of computing the shortest path between a pair of locations are pervasive in Computer Science and Operations Research. Typically, the measures that we optimize are in terms of "distance" traveled, or time spent (or in some cases, a combination of the two). There are literally thousands of papers dealing with problems related to shortest-path and tour problems.

In this paper, we consider a more general model that incorporates the actual cost in terms of *gas prices*. We have a vehicle with a given tank capacity of U. In fact, we will assume that U is the distance the vehicle may travel on a full tank of gas (this can easily be obtained by taking the product of the tank size and the mileage per gas unit of the vehicle). Moreover, we may assume that we start with some given amount of gas μ ($\leq U$) in the tank. We assume that at each vertex v gas may be purchased at a price of $c(v)$. This price is the cost of gas per mile. For example if gas costs \$3.40 per gallon and the vehicle can travel for 17 miles per gallon, then the cost per mile is 20 cents.

* Research supported by NSF grant CCF-0430650.
Full version available at http://www.cs.umd.edu/∼samir/grant/esa07.ps

At each gas station we may fill up some amount of gas to "extend" the range of the vehicle by a certain amount. Moreover, since gas prices vary, the cost depends on where we purchase gas from.

In addition to fluctuating gas prices, there is significant variance in the price of gas between gas stations in different areas. For example, in the Washington DC area alone, the variance in gas prices between gas stations in different areas (on the same day) can be by as much as 20%. Due to different state taxes, gas prices in adjacent states also vary. Finally, one may ask: why do we expect such information to be available? In fact, there are a collection of web sites [1, 2] that currently list gas prices in an area specified by zip code. So it is reasonable to assume that information about gas prices is available. What we are interested in are algorithms that will let us compute solutions to some basic problems, given this information.

In this general framework, we are interested in a collection of basic questions.

1. (The gas station problem) Given a start node s and a target node t, how do we go from s to t in the cheapest possible way if we start at s with μ_s amount of gas? In addition we consider the variation in which we are willing to stop to get gas at most Δ times[1]. Another generalization we study is the sequence gas station problem. Here, we want to find the cheapest route that visits a set of p locations in a specified order (for example by a delivery vehicle).
2. (The fixed-path gas station problem) An interesting special case is when we fix the path along which we would like to travel. Our goal is to find an optimal set of refill stops along the path.
3. (The uniform cost tour gas station problem) Given a collection of cities T, and a set of gas stations S at which we are willing to purchase gas, find the shortest tour that visits T. We have to ensure that we never run out of gas. Clearly this problem generalizes the Traveling Salesman Problem. The problem gets more interesting when $S \neq T$, and we address this case. This models the situation when a large transportation company has a deal with a certain gas company, and their vehicles may fill up gas at any station of this company at a pre-negotiated price. Here we assume that gas prices are the same at each gas station. This could also model a situation where some gas stations with very high prices are simply dropped from consideration, and the set S is simply the set of gas stations that we are willing to use.
4. (The tour gas station problem) This is the same as the previous problem, except that the prices at different stations can vary.

Of all the above problems, only the tour problems are NP-hard. For the first two we develop polynomial time algorithms, and for the tour problems we develop approximation algorithms.

[1] This restriction makes sense, because in some situations where the gas prices are decreasing as we approach our destination, the cheapest solution may involve an arbitrarily large number of stops, since we only fill up enough gas to make it to a cheaper station further down the path.

We now give a short summary of the results in the paper:

1. (The gas station problem) For the basic gas station problem, our algorithm runs in time $O(\Delta n^2 \log n)$ and computes an optimal solution. If we want to visit a sequence of p cities we can find an optimal solution in time $O(\Delta (np)^2 \log(np))$. In addition, we develop a second algorithm for the all-pairs version that runs in time $O(n^3 \Delta^2)$. This method is better than repeating the fixed-destination algorithm n times when $\Delta < \log n$.
2. (The fixed-path gas station problem) For the fixed-path version with an unbounded number of stops, we develop a fast $O(n \log n)$ time algorithm. Due to space constraints this is described in the full version of the paper.
3. (The uniform cost tour gas station problem) Since this problem is NP-hard, we focus on polynomial time approximation algorithms. We assume that every city has a gas station within a distance of $\alpha \frac{U}{2}$ for some $\alpha < 1$. This assumption is reasonable since in any case, every city has to have a gas station within distance $\frac{U}{2}$, otherwise there is no way to visit it. A similar assumption is made in the work on distance constrained vehicle routing problem [13]. We develop an approximation algorithm with an approximation factor of $\frac{3}{2}(\frac{1+\alpha}{1-\alpha})$. We also consider a special case, namely when there is only one gas station. This is the same as having a central depot, and requiring the vehicle to return to the depot after traveling a maximum distance of U. For this special case, we develop an algorithm with factor $O(\ln \frac{1}{1-\alpha})$ and this improves the bound of $\frac{3}{2(1-\alpha)}$ given by Li et al. [13] for the distance constrained vehicle routing problem.
4. (The tour gas station problem) For the tour problem with arbitrary prices, we can use the following scheme: sort all the gas prices in non-decreasing order $c_1 \leq c_2 \leq \ldots c_n$. Now guess a range of prices $[c_i \ldots c_j]$ one is willing to pay, and let $\beta_{ij} = \frac{c_j}{c_i}$. Let S_{ij} include all the gas stations v such that $c_i \leq c(v) \leq c_j$. We can run the algorithm for the *uniform cost tour gas station problem* with set S_{ij} and cities T. This will yield a tour $\mathcal{T}^{[i,j]}$. We observe that the cost of the tour $\mathcal{T}^{[i,j]}$ is at most $O(\frac{\beta_{ij}}{1-\alpha})$ times the cost of an optimal solution, since its possible that we always pay a factor β_{ij} more than the optimal solution, at each station where we fill gas. Taking the best solution over all $O(n^2)$ possible choices gives a valid solution to the tour gas station problem.

1.1 Related Work

The problems of computing shortest paths and the shortest TSP tour are clearly the most relevant ones here and are widely studied, and discussed in several books [12, 17].

One closely related problem is the Orienteering problem [4, 5, 10, 7]. In this problem the goal is to compute a path of a fixed length L that visits as many locations as possible, starting from a specified vertex. For this problem, a factor 3 approximation has been given recently by Bansal et al. [6]. (In fact, they can fix the starting and ending vertices.) This algorithm is used as subroutine for developing a bicriteria bound for Deadline TSP. By using the 3 approximation for the Orienteering problem, we develop an $O(\log |T|)$ approximation for the single gas station tour problem. This is not surprising, since we would like to cover all the locations by finding walks of length at most U.

There has been some recent work by Nagarajan and Ravi [16] on minimum vehicle routing that is closely related to the single gas station tour problem. In this problem, a designated root vertex (depot) and a deadline D are given and the goal is to use the minimum number of vehicles from the root so that each location is met by at least one of the vehicles, and each vehicle traverses length at most D. (In their definition, vehicles do not have to go back to the root.) They give a 4-approximation for the case where locations are in a tree and an $O(\log D)$ approximation for graphs with integer weights.

Another closely related piece of work is by Arkin et al. [3] where tree and tour covers of bounded length are computed. What makes their problem easier is that there is no specified root node, or a set of gas stations one of which should be included in any bounded length tree or tour. Several pieces of work deal with vehicle routing problems [14, 15, 9] with multiple vehicles, where the objective is to bound the total cost of the solution, or to minimize the longest tour. However these problems are significantly easier to develop approximation algorithms for.

2 The gas station problem

The input to our problem consists of a complete graph $G = (V, E)$ with edge lengths $d : E \to R^+$, gas costs $c : V \to R^+$ and a tank capacity U. (Equivalently, if we are not given a complete graph we can define d_{uv} to be the distance between u and v in G.) Our goal is to go from a source s to a destination t in the cheapest possible way using at most Δ stops to fill gas. For ease of exposition we concentrate on the case where we start from s with an empty tank. The case in which we start with μ_s units of gas can be reduced to the former as follows. Add a new node s' such that $d_{s's} = U - \mu_s$ and $c(s') = 0$. The problem of starting from s with μ_s units of gas and that of starting from s' with an empty tank using one additional stop are equivalent.

We would also like to note that our strategy yields a solution where the gas tank will be empty when one reaches a location where gas can be filled cheaply. In practice, this is not safe and one might run out of gas (for example if one gets stuck in traffic). For that reason we suggest defining U to be *smaller* than the actual tank capacity so that we always have some “reserve” capacity.

In this section we develop an $O(\Delta n^2 \log n)$ time algorithm for the gas station problem. In addition, when $\Delta = n$ we show how to solve the problem in $O(n^3)$ time for general graphs, and $O(n \log n)$ time for the case where G is a fixed path.

One interesting generalization of the problem is the *sequence gas station problem* where we are given a sequence $s_1, s_2, \ldots, s_p$ of vertices that we must visit in the specified order. This variant can be reduced to the s-t version in an appropriately defined graph.

2.1 The gas station problem using Δ stops

We will solve the gas station problem using the following dynamic program (DP) formulation:

$$A(u, q, g) = \begin{array}{l}\text{Minimum cost of going from } u \text{ to } t \text{ using } q \text{ refill stops, starting} \\ \text{with } g \text{ units of gas. We consider } u \text{ to be one of the } q \text{ stops.}\end{array}$$

The main difficulty in dealing with the problem stems from the fact that, in principle, we need to consider every value of $g \in [0, U]$. One way to avoid this is to discretize the values g can take. Unfortunately this only yields a pseudo-polynomial time algorithm. To get around this we need to take a closer look at the structure of the optimal solution.

Lemma 1. *Let $s = u_1, u_2, \ldots, u_l$ be the refill stops of an optimal solution using at most Δ stops. The following is an optimal strategy for deciding how much gas to fill at each stop: At u_l fill just enough to reach t with an empty tank; for $j < l$*

i) If $c(u_j) < c(u_{j+1})$ then at u_j fill up the tank.
ii) If $c(u_j) \geq c(u_{j+1})$ then at u_j fill just enough gas to reach u_{j+1}.

Proof. If $c(u_j) < c(u_{j+1})$ and the optimal solution does not fill up at u_j then we can increase the amount filled at u_j and decrease the amount filled at u_{j+1}. This improves the cost of the solution, which contradicts the optimality assumption. Similarly, if $c(u_j) \geq c(u_{j+1})$ then we can decrease the amount filled at u_j and increase the amount filled at u_{j+1} (without increasing the overall cost of the solution) until the condition is met. ☐

Consider a refill stop $u \neq s$ in the optimal solution. Let w be the stop right before u. Lemma 1 implies that if $c(w) > c(u)$, we reach u with an empty tank, otherwise we reach u with $U - d_{wu}$ gas. Therefore, in our DP formulation we need to keep track of at most n different values of gas for u. Let $GV(u)$ be the set of such values, namely

$$GV(u) = \{U - d_{wu} \mid w \in V \text{ and } c(w) < c(u) \text{ and } d_{wu} \leq U\} \cup \{0\}$$

The following recurrence allows us to compute $A(u, q, g)$ for any $g \in GV(u)$:

$$A(u, 1, g) = \begin{cases} (d_{ut} - g)\, c(u) & \text{if } g \leq d_{ut} \leq U \\ \infty & \text{otherwise} \end{cases}$$

$$A(u, q, g) = \min_{\substack{v \text{ s.t.} \\ d_{uv} \leq U}} \left\{ \begin{array}{ll} A(v, q-1, 0) + (d_{uv} - g)\, c(u) & \mid c(v) \leq c(u) \wedge g \leq d_{uv} \\ A(v, q-1, U - d_{uv}) + (U - g)\, c(u) & \mid c(v) > c(u) \end{array} \right\}$$

The cost of the optimal solution is $\min_{1 \leq l \leq \Delta} A(s, l, 0)$. The naive way of filling the table takes $O(\Delta n^3)$ time. However, this can be done more efficiently.

Theorem 1. *There is an $O(\Delta n^2 \log n)$ time algorithm for the gas station problem with Δ stops.*

Instead of spending $O(n)$ time computing a single entry of the table, we spend $O(\log n)$ amortized time per entry. More precisely, for fixed $u \in V$ and $1 < q \le \Delta$ we show how to compute all entries of the form $A(u, q, *)$ in $O(n \log n)$ time using entries of the form $A(*, q-1, *)$. Theorem 1 follows immediately.

The DP recursion for $A(u, q, g)$ finds the minimum, over all v such that $d_{uv} \le U$, of terms that corresponds to the cost of going from u to t through v. Split each of these terms into two parts based on whether they depend on g or not. Thus we have an independent part, which is either $A(v, q-1, 0) + d_{uv}\, c(u)$ or $A(v, q-1, U - d_{uv}) + U c(u)$; and a dependent part, $-g\, c(u)$.

Our procedure begins by sorting the independent part of every term. Note that the minimum of these corresponds to the entry for $g = 0$. As we increase g, the terms decrease uniformly. Thus, to compute the table entry for $g > 0$ just subtract $g\, c(u)$ from the smallest independent part available. The only caveat is that the term corresponding to a vertex v such that $c(v) \le c(u)$ should not be considered any more once $g > d_{uv}$, we say such a term *expires* after $g > d_{uv}$. Since the independent terms are sorted, once the smallest independent term expires we can walk down the sorted list to find the next vertex which has not yet expired. The procedure is dominated by the time spent sorting the independent terms which takes $O(n \log n)$ time.

Theorem 2. *When $\Delta = n$ the problem can be solved in $O(n^3)$ time.*

We can reduce the problem to a shortest path question on a new graph H. The vertices of H are pairs (u, g), where $u \in V$ and $g \in GV(u)$. The edges of H and their weight $w(\cdot)$ are defined by the DP recurrence. Namely, for every $u, v \in V$ and $g \in GV(u)$ such that $d_{uv} \le U$ we have $w\big((u, q), (v, 0)\big) = (d_{uv} - g)\, c(u)$ if $c(v) \le c(u)$ and $g \le d_{uv}$, or $w\big((u, q), (v, U - d_{uv}\big) = (U - g)\, c(u)$ if $c(v) > c(u)$.

Our objective is to find a shortest path from $(s, 0)$ to $(t, 0)$. Note that H has at most n^2 vertices and at most n^3 edges. Using Dijkstra's algorithm [8] the theorem follows.

2.2 Faster algorithm for the all-pairs version

Consider the case in which we wish to solve the problem for all starting nodes i, with μ_i amount of gas in the tank initially. Using the method described in the previous section, we get a running time of $O(n^3 \Delta \log n)$ since we run the algorithm for each possible destination. We will show that for $\Delta < \log n$ we can improve this and get a bound of $O(n^3 \Delta^2)$.

Add new nodes i' such that $d_{i'i} = U - \mu_i$ and $c(i') = 0$. If we start at i with μ_i units of gas, it is the same as starting from i' where gas is free. We fill up the tank to capacity U, and then by the time we reach i we will have exactly μ_i units of gas in the tank. (Since gas is free at any node i' in any optimal solution we fill up the tank to capacity U). This will use one extra stop.

We define $B(i, h, p)$ as the minimum cost solution to go from i to h (destination), with p stops to get gas, given that we start with an empty tank at i. Since we start with an empty tank, we have to fill up gas at the starting point (and this is included as one of the stops). Clearly, we will also reach h (destination) with an empty tank, assuming that there is no trivial solution, such as one that arrives at the destination with no fill-ups on the way.

Our goal is to compute $B(i', h, \Delta+1)$ which is a minimum cost solution to go from i' to h with at most Δ stops in-between. Note that the first fill-up is the one that takes place at node i', after that we stop at most Δ times.

We will now show how to compute $B(i, h, p)$. There are two options:

- If the gas price at the first stop after i (e.g. k) is cheaper than $c(i)$ then we will reach that station with an empty tank after filling d_{ik} units of gas at i (as long as $d_{ik} \leq U$):

$$B(i, h, p) = B(k, h, p-1) + d_{ik}c(i)$$

- If the first place where the cost of gas decreases from the previous stop is the $q+1^{st}$ stop and the price is in increasing order in the first q stops then

$$B(i, h, p) = C(i, k, q) + B(k, h, p-q)$$

  We define $C(i, k, q)$ as the minimum cost way of going from i to k with at most q stops to get gas, such that we start at i with an empty tank (and get gas at i, which counts as a stop) and finally reach k with an empty tank. In addition, the price of gas in intermediate stations is in increasing order except for the last stop.

We define $B(h, h, p) = 0$. For $i \neq h$ let $B(i, h, 1) = c(i)\, d_{ih}$ if $d_{ih} \leq U$, and $B(i, h, 1) = \infty$ otherwise. In general:

$$B(i,h,p) = \min \left\{ \min_{\substack{1 \leq k \leq n \\ 1 < q \leq p}} C(i,k,q) + B(k,h,p-q), \min_{\substack{1 \leq k \leq n \\ \text{s.t.}\, d_{ik} \leq U}} B(k,h,p-1) + d_{ik}\, c(i) \right\}$$

If we are able to compute $C(i, k, q)$ efficiently then $B(i, h, p)$ can be computed. There are $n^2\Delta$ states in the dynamic program, and each one can be computed in time $O(n\Delta)$. This yields a running time of $O(n^3\Delta^2)$. We will see that the time required to compute $C(i, k, q)$ is $O(n^3\Delta)$ for all relevant choices of i, k, q.

Suppose that in going from i to k we stop at $i_1 = i, \ldots, i_q, i_{q+1} = k$. Note that $c(i_1) \leq c(i_2) \leq \ldots \leq c(i_q)$, however $c(i_q) > c(i_{q+1})$. In fact, at i_1 we will get U amount of gas. When we reach i_j for $1 < j < q$, we will get $d_{i_{j-1}i_j}$ units of gas (the amount that we consumed since the previous fill-up) at a cost of $c(i_j)$ per unit of gas. The amount of gas we will get at i_q is just enough to reach k with an empty tank. Now we can see that the total cost is equal to $Uc(i_1) + d_{i_1 i_2}c(i_2) + \ldots + d_{i_{q-2}i_{q-1}}c(i_{q-1}) + (d_{i_{q-1}i_q} + d_{i_q k} - U)c(i_q)$. Note that the last term is not negative, since we could not reach k from i_{q-1} even with a full tank at i_{q-1}, without stopping to get a small amount of gas.

We compute $C(i, k, q)$ as follows. First note that if $d_{ik} \leq U$ then the answer is $d_{ik}c(i)$. Otherwise we build a directed graph $G' = (V \cup V_D, E \cup E_D)$, where V is the set of vertices, and $V_D = \{i' | i \in V\}$.

We define E: add a directed edge from $i \in V$ to j for each vertex $j \in V \setminus \{i\}$ such that $d_{ij} \leq U$ and $c(i) \leq c(j)$. The weight of this edge is $d_{ij}c(j)$.

We define E_D as follows: add a directed edge from each $j \in V$ to k' for each vertex $k' \in V_D \setminus \{j'\}$ such that $U < d_{jk} \leq 2U$. The weight of this edge is

$$\min \left\{ (d_{jz} + d_{zk} - U)c(z) \,|\, c(j), c(k) < c(z) \text{ and } d_{jz}, d_{zk} \leq U \right\}$$

Now we can express $C(i,k,q)$ as $Sp(i,k',q) + Uc(i)$ where $Sp(i,k',q)$ is the shortest path from i to k' in the graph G' using at most q edges.

To see why it is true, we can see that for any given order of stops between i and k (where the gas price is in increasing order in consecutive stops), the minimum cost is equal to the weight of the path in G' that starts from i, goes to the second stop in the given order (e.g., i_2) and then traverses the vertices of V in the same order and from the second last stop goes to k'. It is also possible that $q = 2$ and the path goes directly from $i = i_1$ to k in this case, and i_2 is the choice for z that achieves the minimum cost for the edge (i, k').

For any given path P in G' between i and k', if the weight of the path is W_P we can find a feasible plan for filling the tank at the stations so that the cost is equal to $W_P + Uc(i)$. It is enough to fill up the tank at the stations that are in the path, except the last one in which the tank is filled to only the required level to reach k. We can conclude that $C(i,k,q)$ is equal to $Sp(i,k',q) + Uc(i)$.

The running time for finding the shortest path between all pairs of nodes with different number of stops (at most Δ) can be computed in $O(n^3\Delta)$ by dynamic programming [11]. If we precompute $C(i,k,q)$ the running time for computing $B(i', h, \Delta+1)$ is $O(n^3\Delta^2)$ assuming we start at i with μ_i amount of gas. So in general the running time is $O(n^3\Delta^2)$.

3 The uniform cost tour gas station problem

In this section we study a variant of the gas station problem where we must visit a set of cities T in arbitrary order. We consider the case where gas costs the same at every gas station, but some cities may not have a gas station.

More formally, the input to our problem consists of a complete undirected graph $G = (V, E)$ with edge lengths $d : E \rightarrow R^+$, a set of cities $T \subseteq V$, a set of gas stations $S \subseteq V$, and tank capacity U for our vehicle. The objective is to find a minimum length tour that visits all cities in T, and possibly some gas stations in S. We are allowed to visit a location multiple times if necessary. We require any segment of the tour of length U to contain at least one gas station, this ensures we never run out of gas. We call this the *uniform cost tour gas station problem*. We assume that we start with an empty tank at a gas-station.

The problem is NP-hard as it generalizes the well-known traveling salesman problem: just set the tank capacity to the largest distance between any two cities and let $T = S$. In fact, there is a closer connection between the two problems: If every city has a gas station, i.e., $T \subseteq S$, we can reduce the gas station problem to the TSP. Consider a TSP instance on T under metric $\ell : T \times T \rightarrow R^+$, where ℓ_{xy} is the minimum cost of going between cities x and y starting with an empty tank (this can be computed by standard techniques). Since the cost of gas is the same everywhere, a TSP tour can be turned into a driving plan that visits all cities with the same cost and vice-versa. Let OPT denote an optimal solution, and $c(OPT)$ its cost.

As mentioned earlier, we can use the algorithm for the uniform cost case to derive an approximation algorithm for the general case by paying a factor β in the approximation ratio. Here β is the ratio of the maximum price that an optimal solution pays for buying a unit of gas, to the minimum price it pays for buying a unit of gas (in practice this ranges from 1 to 1.2).

Unfortunately this reduction to the TSP breaks down when cities are not guaranteed to have a gas station. Consider going from x to y, where x does not have a gas station. The distance between x and y will depend on how much gas we have at x, which in turn depends on which city was visited before x and what route we took to get there.

An interesting case of the tour gas station problem is that of an instance with a single gas station. This is also known as the *distance constrained vehicle routing problem* and was studied by Li et al. [13] who gave a $\frac{3}{2(1-\alpha)}$ approximation algorithm, where the distance from the gas station to the most distant city is $\alpha\frac{U}{2}$, for some $\alpha < 1$. We improve this by providing an $O(\log \frac{1}{1-\alpha})$ approximation algorithm Without making any assumptions on α we show that a greedy algorithm that finds bounded length tours visiting the most cities at a time is a $O(\log |T|)$-factor approximation. The proof of these claims appear in the full version of this paper.

For the general case we make the assumption that every city has a gas station at distance at most $\alpha\frac{U}{2}$. This assumption is reasonable, because if a city has no gas station within distance $\frac{U}{2}$, there is no way to visit it. We show a $\frac{3(1+\alpha)}{2(1-\alpha)}$ approximation for this problem. Note that when $\alpha = 0$, this gives the same bound as the Christofides method for the TSP.

3.1 The tour gas station problem

For each city $x \in T$ let $g(x) \in S$ be the closest gas station to x, and let d_x be the distance from x to $g(x)$. We assume that every city has a gas station at distance at most $\alpha\frac{U}{2}$; in order words, $d_x \leq \alpha\frac{U}{2}$ for all $x \in T$.

Recall that it is assumed that the price of the gas is the same at all the gas stations. We define a new distance function for the distance between each pair of cities. The distance ℓ is defined as follows: For each pair of cities x and y, ℓ_{xy} is the length of the shortest traversal to go from x to y starting with $U - d_x$ amount of gas and reaching y with d_y amount of gas. If $d_{xy} \leq U - d_x - d_y$ then we can go directly from x to y, and $\ell_{xy} = d_{xy}$. Otherwise, we can compute this as follows. Create a graph whose vertex set is S, the set of gas stations. To this graph add x and y. We now add edges from x to all gas stations within distance $U - d_x$ from x. Similarly we add edges from y to all gas stations within distance $U - d_y$ to y. Between all pairs of gas stations, we add an edge if the distance between the pair of gas stations is at most U. All edges have length equal to the distance between their end points. The length of the shortest path in this graph from x to y will be ℓ_{xy}. Note that the shortest path (in general) will start at x and then go through a series of gas stations before reaching y. This path yields a valid plan to drive from x to y without running out of gas, once we reach x with $U - d_x$ units of gas. When we reach y, we have enough gas to go to g_y. Also note that $\ell_{xy} = \ell_{yx}$ since the path is essentially "reversible".

We assume here that all distances are Euclidean. Note that from x, we can only go to B and not A since we start from x with $U - d_x$ units of gas. From B, we cannot go to D since the distance between B and D is more than U, even though the path through D to y would be shorter. From C we go to E since going through F will give a longer path, since from F we cannot go to y directly.

Note that the function ℓ may not satisfy triangle inequality. To see this, suppose we have three cities x, y, z. Let $d_{xy} = d_{yz} = \frac{U}{2}$. Let $d_x = d_y = d_z = \frac{U}{4}$ and $d_{xz} = U$. We first observe that $\ell_{xy} = \ell_{yz} = \frac{U}{2}$. However, if we compute ℓ_{xz}, we cannot go from x to z directly since we only have $\frac{3}{4}U$ units of gas when we start at x and need to reach z with $\frac{U}{4}$ units of gas. So we have to visit g_y along the way, and thus $\ell_{xz} = \frac{3}{2}U$.

The algorithm is as follows:

1. Create a new graph G', with a vertex for each city. For each pair of cities x, y compute ℓ_{xy} as shown earlier.
2. Find the minimum spanning tree in (G', ℓ). Also find a minimum weight perfect matching M on the odd degree vertices in the MST. Combine the MST and M to find an Euler tour $\mathcal{T}$.
3. Start traversing the Eulerian tour. Add refill trips whenever needed. (Details on this follow.)

It can be shown that the total length of the MST is less than the optimal solution cost. Suppose $x_1, \ldots, x_n$ is the order in which the optimal solution visits the cities. Clearly, the cost of going from x_i to x_{i+1} in the optimal solution is at least $\ell_{x_i x_{i+1}}$. Since the collection of edges (x_i, x_{i+1}) forms a spanning tree, we can be conclude that the weight of the $\ell(\text{MST}) \leq c(OPT)$. Next we show that the cost of M is at most $\frac{c(OPT)}{2}$. Suppose the odd degree vertices are in the optimal solution in the order $o_1, \ldots, o_k$. We can see that $\ell_{o_i o_{i+1}}$ is at most equal to the distance we travel in the optimal solution to go from o_i to o_{i+1}. So the cost of minimum weighted matching on the odd degree vertices is at most $\frac{c(OPT)}{2}$. So the total cost of the Eulerian tour $\mathcal{T}$ is at most $\frac{3c(OPT)}{2}$.

Now we need to transform the Eulerian tour into a feasible plan. First, every edge (x, y) in $\mathcal{T}$ is replaced with the actual plan to drive from x to y that we found when computing ℓ_{xy}. If $d_{xy} \leq U - d_x - d_y$ the plan is simply to go straight from x to y, we call these *direct edges*. Otherwise the plan must involve stopping along the way in one or more gas stations, we call these *indirect edges*. Notice that the cost of this plan is exactly that of the Eulerian tour $\mathcal{T}$. Unfortunately, as we will see below this plan need not be feasible.

Define a strand, to be a sequence of consecutive cities in the tour connected by direct edges. If a city is connected with two indirect edges, then it forms a

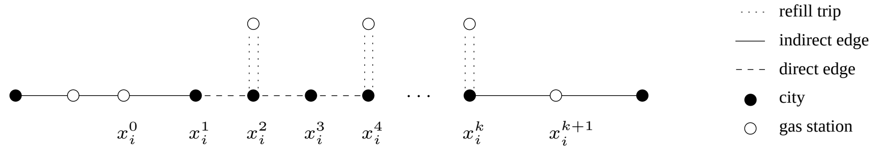

Fig. 1. Decomposition of the solution into strands.

strand by itself. Suppose the i^{th} strand has cities $x_i^1, \ldots, x_i^k$. To this we add x_i^0 (x_i^{k+1}), the last (first) gas station in the indirect edge connecting x_i^1 (x_i^k) with the rest of the tour. Each strand now starts and ends with a gas station. We can view the tour as a decomposition into strands as shown in Fig. 1. Note that if the distance between x_i^0 and x_i^{k+1} is more than U the overall plan is not feasible. To fix we add for every city a refill trip to its closest gas station and then greedily try to remove them, while maintaining feasibility, until we get a minimal set of refill trips. Let us bound the extra cost these trips incur.

Lemma 2. *Let L_i be the length of the ith strand. Then the total distance traveled on the refill trips of cities in the strand is at most $\frac{2\alpha}{1-\alpha}L_i$.*

Proof. Assume there are q_i refill trips in this strand. Label the cities with refill trips to their nearest gas stations $x_i^{j_1}, \ldots, x_i^{j_{q_i}}$. Also label x_i^0 as $x_i^{j_0}$ and x_i^k as $x_i^{j_{q_i+1}}$. Note that $\ell(\mathcal{T}(x_i^{j_p}, x_i^{j_{p+2}})) \geq (1-\alpha)U$ (otherwise the refill trip at $x_i^{j_{p+1}}$ can be dropped). This gives us:

$$2L_i \;>\; \sum_{0\leq p\leq q_i-1} \ell(\mathcal{T}(x_i^{j_p}, x_i^{j_{p+2}})) \;\geq\; q_i(1-\alpha)U \;\Longrightarrow\; q_i \leq \frac{2L_i}{(1-\alpha)U}$$

The length of each refill trip no more than αU. Therefore, the total length of the refill trips is at most $\alpha U q_i$, and the lemma follows. ☐

The cost of the solution is the total length of the strands (which is the length of the tour) plus the total cost of the refill trips. (Note that without loss of generality we can assume that our tour always starts from a gas station. For the case with only direct edges, there is exactly one strand, starting and ending at the first city with the gas station).

In other words, the total cost of the solution is:

$$\ell(\mathcal{T}) + \sum_i \alpha U q_i \leq \left(1 + \frac{2\alpha}{1-\alpha}\right)\ell(\mathcal{T}) \leq \left(\frac{1+\alpha}{1-\alpha}\right)\frac{3}{2}\,c(\mathrm{OPT}).$$

Theorem 3. *There is a $\frac{3\,(1+\alpha)}{2\,(1-\alpha)}$-approximation for the tour gas station problem.*

4 Conclusions

Current problems of interest are to explore improvements in the approximation factors for the special cases of Euclidean metrics, and planar graphs. In addition we would also like to develop faster algorithms for the single source and destination case, perhaps at the cost of sacrificing optimality of the solution.

References

1. `http://www.gasbuddy.com/`.
2. `http://www.aaa.com/`.
3. E. M. Arkin, R. Hassin, and A. Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006.
4. E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of the 14th Annual Symposium on Computational Geometry (SoCG)*, pages 307–316, 1998.
5. B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing*, 28(1):254–262, 1998.
6. N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the 36th annual ACM symposium on Theory of computing (STOC)*, pages 166–174, 2004.
7. A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. In *Proceedings of the 44rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, page 46, 2003.
8. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. M.I.T. Press and McGraw-Hill, 2001.
9. G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.
10. B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.
11. E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Publications, 2001.
12. E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.
13. C.-L. Li, D. Simchi-Levi, and M. Desrochers. On the distance constrained vehicle routing problem. *Operations Research*, 40(4):790–799, 1992.
14. A. R. K. M. Haimovich. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.
15. L. S. M. Haimovich, A.G. Rinnoooy Kan. Analysis of heuristics for vehicle routing problems. *Vehicle Routing: Methods and Studies*, pages 47–61, 1988.
16. V. Nagarajan and R. Ravi. Minimum vehicle routing with a common deadline. In *Proceedings of the 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 212–223, 2006.
17. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Dover Publications, Inc., 1998.