Fly Cheaply: On the Minimum Fuel Consumption Problem

Timothy M. Chan* Alon Efrat† Sariel Har-Peled‡

September 30, 1999

Abstract

In planning a flight, stops at intermediate airports are sometimes necessary to minimize fuel consumption, even if a direct flight is available. We investigate the problem of finding the cheapest path from one airport to another, given a set of n airports in $\mathbb{R}^2$ and a function $l : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^+$ representing the cost of a direct flight between any pair.

Given a source airport s, the cheapest-path map is a subdivision of $\mathbb{R}^2$ where two points lie in the same region iff their cheapest paths from s use the same sequence of intermediate airports. We show a quadratic lower bound on the combinatorial complexity of this map for a class of cost functions. Nevertheless, we are able to obtain subquadratic algorithms to find the cheapest path from s to all other airports for any well-behaved cost function l: our general algorithm runs in $O(n^{4/3+\varepsilon})$ time, and a simpler, more practical variant runs in $O(n^{3/2+\varepsilon})$ time, while a special class of cost functions requires just $O(n \log n)$ time.

1 Introduction

Assume we wish to plan a path of a flight from a starting airport s to a destination airport t. Our plane is able to carry enough fuel, so a direct flight from s to t is possible. However, we might find it cheaper to stop at some airports, even if this increases the total length of our path. One reason to do so is that the longer we fly without a stop, the more fuel we need to take, so the weight of the plane increases, and the cost of a mile of flight increases. In this paper, we investigate the problem of how to plan the most economical path (i.e., the cheapest).

We formalize the problem as follows. Let P be a set of n points in $\mathbb{R}^2$, containing two special points s and t. Also given is a positive function $l : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^+$, where

*Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada; *tmchan@math.uwaterloo.ca*; *http://www.cs.uwaterloo.ca/~tmchan*

†Computer Science Department, Stanford University, Stanford, CA 94305, USA; *alon@cs.stanford.edu*; *http://graphics.stanford.edu/~alon*

‡School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel; *sariel@math.tau.ac.il*; *http://www.math.tau.ac.il/~sariel/*

$l(p, q)$ represents the cost of the flight from p to q for any pair $p, q \in P$. We call a sequence $\pi = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0, v_1, \ldots, v_{k-1} \in P$, a *path* from v_0 to v_k. The *cost* of π is the sum $\sum_{i=1}^{k} l(v_{i-1}, v_i)$. The *cheapest-path problem* is to find a path from s to t of the minimum cost. We denote this cheapest path by $\pi^*(t)$ and its cost by $d[t]$.

In this paper, we assume that the function l is of *constant descriptive complexity*, i.e., it is continuous, total, and piecewise-algebraic, defined by a constant number of polynomials with a constant maximum degree.

Due to the aforementioned motivation, we are primarily interested in cost functions that are *monotone* in the following sense: $l(p, q) < l(p, q')$ iff $|pq| < |pq'|$, where $|pq|$ stands for the Euclidean distance between $p, q \in \mathbb{R}^2$.

Clearly, we can solve the cheapest-path problem in time $O(n^2)$ by computing the value $l(p, q)$ for every pair $p, q \in P$ and applying Dijkstra's algorithm on the complete graph. In this paper we show that we can do better.

Section 2 describes several algorithms for the cheapest-path problem. It presents an $O(n \log n)$-time algorithm in the special case where $l(p, q) = |pq|^2 r(|pq|)$ for some monotone increasing function r. Regarding the general case, Section 2 presents an $O(n^{4/3+\varepsilon})$-time[1] algorithm that uses a (relatively complicated) dynamic data structure of Agarwal et al. [AES95]. A more practical $O(n^{3/2+\varepsilon})$-time algorithm is also described (assuming monotonicity) that uses much simpler data structures.

It turns out that the running time of both algorithms is related to the combinatorial complexity of a geometric construct that we will call the *cheapest-path map*: given point set P, cost function l, and source point s, the map $\mathcal{M}$ is defined as the planar subdivision where two points $q, q' \in \mathbb{R}^2$ are assigned the same region iff the cheapest paths $\pi^*(q)$ and $\pi^*(q')$ share the same sequence of intermediate points of P. The map $\mathcal{M}$ can be expressed as the planar projection of the lower envelope of the functions $\{f_i(q) = d[p_i] + l(p_i, q) \mid p_i \in P\}$, thus it follows from a theorem by Halperin and Sharir [HS94] that the complexity of $\mathcal{M}$ is $O(n^{2+\varepsilon})$. However, since the functions discussed above are of restricted type, one might wonder if it is possible to show that the actual complexity of $\mathcal{M}$ is smaller. Section 3 shows that the answer is negative: for a natural class of monotone cost functions l, the complexity of $\mathcal{M}$ can indeed be $\Omega(n^2)$. This lower bound indicates the difficulty in improving the analysis of both our subquadratic algorithms in the general case.

[1] Throughout this paper, ε stands for an arbitrarily small positive constant; the constants of proportionality in bounds may depend on ε.

2 Algorithms for the Cheapest-Path Problem

2.1 Superquadratic Cost Functions

In this subsection, we present a simple and efficient algorithm for a *superquadratic* cost function, i.e., $l(p,q) = |pq|^2 r(|pq|)$, where $r : \mathbb{R}^+ \to \mathbb{R}^+$ is a monotone increasing function (computable in constant time).

Theorem 2.1 *For l superquadratic, the cheapest-path problem can be solved in time $O(n \log n)$.*

Proof: The Delaunay triangulation of the point set P is known to contain the *Gabriel graph* [GO97], defined as the graph G of all edges (p,q) such that the disk with diameter $\overline{pq}$ contains no points of P in its interior ($p, q \in P$). We claim that the edges of any cheapest path must belong to G. Consequently, as the Delaunay triangulation has linear size, we can solve the cheapest-path problem by computing the triangulation in $O(n \log n)$ time [GO97] and applying Dijkstra's algorithm on the triangulation (under cost function l) in $O(n \log n)$ time.

To verify the claim, consider an edge (p,q) not in G. The open disk with diameter $\overline{pq}$ contains some other point, say $v \in P$. Then $|pv|^2 + |vq|^2 < |pq|^2$, implying that

$$\begin{aligned} l(p,v) + l(v,q) &= |pv|^2 r(|pv|) + |vq|^2 r(|vq|) \\ &\leq |pv|^2 r(|pq|) + |vq|^2 r(|pq|) \\ &< |pq|^2 r(|pq|) \;=\; l(p,q). \end{aligned}$$

So, (p,q) cannot be an edge of any cheapest path, since the subpath $\langle p, v, q \rangle$ costs less. □

2.2 General Cost Functions

In this subsection, we present an $O(n^{4/3+\varepsilon})$-time algorithm for general cost functions (of constant descriptive complexity). The idea is to speed up Dijkstra's algorithm using geometric data structures.

Instead of the common form of Dijkstra's algorithm, the following alternative form (for example, see [Baa88]) turns out to be more illuminating. This simple procedure computes the cost $d[t]$ of the cheapest path $\pi^*(t)$. A standard modification can output $\pi^*(t)$ as well.

$d[s] = 0$; $S = \{s\}$; $T = P \setminus \{s\}$.
While $t \notin S$ Do
Begin
  Choose a pair $(u, v) \in S \times T$ that minimizes $d[u] + l(u, v)$.
  Set $d[v] = d[u] + l(u, v)$.
  Insert v to S and delete v from T.
End

The key is to realize that the above really reduces the cheapest-path problem to a *dynamic bichromatic closest-pair problem* if we define our distance function $\delta : S \times T \to \mathbb{R}^+$ to be

$$\delta(u, v) = d[u] + l(u, v).$$

(Think of each $u \in S$ as having an additive weight $d[u]$.) In this dynamic problem, we want to maintain a pair $(u, v) \in S \times T$ minimizing $\delta(u, v)$, subject to insertions and deletions on the two sets S and T.

A general technique of Eppstein [Epp95], which we state as a lemma below, handles precisely this task and further reduces the problem to *dynamic nearest neighbors*: (i) build a data structure to find $q \in T$ minimizing $\delta(p, q)$ for a given p, subject to insertions and deletions on T, and (ii) build a data structure to find $p \in S$ minimizing $\delta(p, q)$ for a given q, subject to insertions and deletions on S.

Lemma 2.2 *For any distance function δ, if the dynamic nearest-neighbor problem can be solved with $O(T(n))$ amortized query, insertion, and deletion time, then the dynamic bichromatic closest-pair problem can be solved with $O(T(n) \log n)$ amortized insertion time and $O(T(n) \log^2 n)$ amortized deletion time.*

In our instance, data structure (i) can be obtained by considering the lower envelope of the bivariate functions $\{g_i(p) = l(p, q_i) \mid q_i \in T\}$, and data structure (ii) can be obtained by considering the lower envelope of the bivariate functions $\{f_i(q) = d[p_i] + l(p_i, q) \mid p_i \in S\}$. Both dynamic data structures can be devised from the work of Agarwal et al. [AES95], which, with the appropriate query/update time tradeoff, achieves $T(n) = O(n^{1/3+\varepsilon})$.

Thus, each iteration of Dijkstra's algorithm is doable in amortized time $O(n^{1/3+\varepsilon})$, and the overall time for the algorithm is $O(n^{4/3+\varepsilon})$.

Theorem 2.3 *For any cost function l of constant descriptive complexity, the cheapest-path problem can be solved in time $O(n^{4/3+\varepsilon})$.*

2.3 General Cost Functions: A More Practical Algorithm

The preceding general algorithm is mainly of theoretical interest, because the dynamic data structures used are quite complicated. In this subsection, we present a

more practical alternative. Though it is slower and runs in time $O(n^{3/2+\varepsilon})$, the description is more self-contained, avoids Agarwal et al.'s structures [AES95] as well as Eppstein's technique [Epp95], and exploits the monotonicity property mentioned in the introduction to simplify parts of the data structures.

First, we introduce a notation: let $\delta(S,T)$ represent the minimum $\delta(u,v)$ over all $u \in S$ and $v \in T$. Recall that in Section 2.2, we have reduced the cheapest-path problem to maintaining $\delta(S,T)$ subject to insertions to S and deletions from T. We now present a direct data structure for this dynamic problem, consisting of the following (where a and b are parameters to be set later):

1. A large subset $S_0 \subseteq S$ so that $|S \setminus S_0| \leq n/a$.
2. A partition of T into b groups $T_1, \ldots, T_b$, each of size at most n/b. We keep a standard (Euclidean) Voronoi diagram for each point set T_j, preprocessed for planar point location.
3. A priority queue Q containing these values:
   (a) $\delta(\{p\}, T_j)$ for each $p \in S \setminus S_0$ and each $j \in \{1, \ldots, b\}$;
   (b) $\delta(S_0, \{q\})$ for each $q \in T$.

It is clear that the minimum of Q is the desired value $\delta(S,T)$.

How do we delete a point q from T? First we locate the group T_j containing q. To delete q from T_j, we have to reconstruct the Voronoi diagram of T_j in $O((n/b)\log n)$ time. We need to delete $\delta(S_0, \{q\})$ from the queue Q. We also need to update the value $\delta(\{p\}, T_j)$ for each $p \in S \setminus S_0$. These at most n/a values can be computed in $O((n/a)\log n)$ time by point location queries [GO97] on the Voronoi diagram of T_j, because the nearest neighbor to p under distance function δ is also the nearest neighbor to p under the Euclidean metric if l is monotone.

How do we insert a point p to S? We insert $\delta(\{p\}, T_j)$ to the queue Q for each $j \in \{1, \ldots, b\}$. These b values can be computed in $O(b \log n)$ time, again by point location queries on the Voronoi diagram of the T_j's.

In addition, we perform a "re-evaluation phase" to reset $S_0 = S$ after every n/a insertions, so that at any time, $|S \setminus S_0|$ is kept below n/a. How is this re-evaluation done? We empty the queue Q and insert $\delta(S_0, \{q\})$ for each $q \in T$. Since $\delta(S, \{q\}) = \min\{\delta(S_0, \{q\}), \delta(S \setminus S_0, \{q\})\}$, it suffices to compute $\delta(S \setminus S_0, \{q\})$ for each $q \in T$. These n values can be found in $O(n \log n)$ time by point location queries on the planar projection of the lower envelope of the bivariate functions $\{f_i(q) = d[p_i] + l(p_i, q) \mid p_i \in S \setminus S_0\}$. This three-dimensional lower envelope of n/a functions has complexity $O((n/a)^{2+\varepsilon})$, as shown by Halperin and Sharir [HS94], and can be constructed and preprocessed in time $O((n/a)^{2+\varepsilon})$, for example, by the algorithm of Agarwal et al. [ASS96].

Thus, the cost of a deletion is $O((n/a + n/b)\log n)$, and the cost of an insertion is $O(b \log n)$. This ignores the cost of the re-evaluation phases, which is $O((n/a)^{2+\varepsilon} + n \log n)$ for every n/a insertions. Choosing $a = b = \lfloor\sqrt{n}\rfloor$, we obtain $O(n^{1/2+\varepsilon})$ amortized update time and an $O(n^{3/2+\varepsilon})$-time algorithm.

Remark 2.4 The preceding algorithms can actually compute the cheapest paths $\pi^*(p)$ (and their costs $d[p]$) from the source s to all points $p \in P$, represented as a cheapest-path tree.

Remark 2.5 The observant reader would recognize, in the algorithms of Sections 2.2 and 2.3, the use of lower envelopes of the bivariate functions $\{f_i(q)\}$, which correspond precisely to cheapest-path maps as defined in the introduction. It is possible to obtain a faster version of either algorithm if the complexity of such a map is subquadratic. Unfortunately, the next section shows that the complexity can be quadratic.

3 Cheapest-Path Maps Can Be Complicated

In this section, we prove that the worst-case complexity of the cheapest-path map $\mathcal{M}$, as defined in the introduction, is $\Omega(n^2)$ for a simple class of superquadratic cost functions: $l(p, q) = |pq|^\alpha$, with $\alpha > 2$.

Theorem 3.1 *For $l(p, q) = |pq|^\alpha$ where $\alpha > 2$ is a constant, there exists a set of $O(n)$ points in the plane so that the cheapest-path map has $\Omega(n^2)$ faces.*

Proof: Choose a sufficiently large parameter A (depending on n) so that

$$(A^2 + 1/4)^{\alpha/2} - A^\alpha > 2n^\alpha, \tag{1}$$

since the limit of the left-hand side is infinite as $A \to \infty$ for $\alpha > 2$. Also choose a sufficiently small parameter $0 < \delta < 1$ (depending on n and A) so that

$$(A + n\delta)^\alpha - A^\alpha < 1, \tag{2}$$

since the left-hand side converges to 0 as $\delta \to 0$.

Our construction is simple:

$$P = \{(0,0), (0,1), (0,2), \ldots, (0,n), (A,0), (A+\delta,0), (A+2\delta,0), \ldots, (A+n\delta,0)\},$$

with $s = (0, 0)$.

To lower-bound the size of the resulting cheapest-path map $\mathcal{M}$, we first introduce a terminology and some observation. We say that a path $\langle v_0, v_1, \ldots, v_k\rangle$ *exits through* v_{k-1}. The same argument used in the proof of Theorem 2.1 implies that a cheapest path $\pi^*(q)$ can only exit through a point p with the property that the disk with

diameter $\overline{pq}$ is free of points of P. In the definition of $\mathcal{M}$, two points $q, q' \in \mathbb{R}^2$ are assigned the same region iff their cheapest paths $\pi^*(q)$ and $\pi^*(q')$ exit through the same point of P.

Thus, let $\mathcal{R}_i$ be the region of points $q \in \mathbb{R}^2$ so that $\pi^*(q)$ exits through the point $(A + i\delta, 0) \in P$, for a given $i \in \{1, \ldots, n-1\}$. We claim that

1. $(A + i\delta,\, j + 1/2) \in \mathcal{R}_i$ for all $j \in \{1, \ldots, n-1\}$.
2. $(x, j) \notin \mathcal{R}_i$ for all $x \in \mathbb{R}$ and $j \in \{2, \ldots, n-1\}$.

Consequently, each region $\mathcal{R}_i$ must have $\Omega(n)$ different connected components, so $\mathcal{M}$ must have $\Omega(n^2)$ components, and the theorem follows.

To verify the first claim, we note that by the empty-disk property, the cheapest path $\pi^*(A + i\delta,\, j + 1/2)$ can exit through three possible points of P: $(0, j)$, $(0, j+1)$, or $(A + i\delta, 0)$. In the first two cases, the cost of the cheapest path would clearly exceed $(A^2 + 1/4)^{\alpha/2}$. However, the path $\langle (0,0), (A,0), (A+\delta,0), \ldots, (A+i\delta,0), (A+i\delta,\, j+1/2)\rangle$ costs only

$$A^\alpha + i\delta^\alpha + (j + 1/2)^\alpha \;<\; A^\alpha + 2n^\alpha \;<\; (A^2 + 1/4)^{\alpha/2},$$

by (1). So, $(A + i\delta,\, j + 1/2) \in \mathcal{R}_i$.

To verify the second claim, we may assume that $A < x < A + n\delta$, because otherwise, the empty-disk property would prevent the cheapest path $\pi^*(x, j)$ to exit through $(A + i\delta, 0)$. Any path from $(0, 0)$ to (x, j) that exits through $(A + i\delta, 0)$ has cost exceeding $A^\alpha + j^\alpha$. However, the path $\langle (0,0), (0,1), \ldots, (0,j), (x,j)\rangle$ costs only

$$j + x^\alpha \;<\; j + (A + n\delta)^\alpha \;<\; j + A^\alpha + 1 \;<\; A^\alpha + j^\alpha,$$

by (2). So, $(x, j) \notin \mathcal{R}_i$ and the proof is complete. □

Remark 3.2 The above result is a bit surprising. We have a point set that induces a cheapest-path tree which is non-self-intersecting and of linear size, but if we extend this to a cheapest-path map for the whole plane, we get a map of quadratic complexity. Despite this lower bound, we have an $O(n \log n)$-time algorithm for the cost function under consideration by Section 2.1.

Remark 3.3 Given parameter α and a planar point set $P = \{p_1, \ldots, p_n\}$ with weights $w_1, \ldots, w_n$, consider the Voronoi diagram under the the distance function $\delta(p_i, q) = |p_i q|^\alpha + w_i$. The above result implies a quadratic worst-case lower bound on the size of this diagram for $\alpha > 2$ by setting $w_i = d[p_i]$. In contrast, the cases $\alpha = 1$ (an additively-weighted Voronoi diagram) and $\alpha = 2$ (a power diagram) are known to have linear complexity [GO97]. We are not aware of any combinatorial results when $1 < \alpha < 2$.

Acknowledgments

The authors wish to thank Pankaj Agarwal, Matya Katz, and Micha Sharir for helpful discussions concerning the problems studied in this paper and related problems.

References

[AES95] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 39–50, 1995.

[ASS96] P. K. Agarwal, O. Schwarzkopf, and M. Sharir. The overlay of lower envelopes and its applications. *Discrete Comput. Geom.*, 15:1–13, 1996.

[Baa88] S. Baase. *Computer Algorithms: Introduction to Design and Analysis.* Addison-Wesley, 2nd ed., 1988.

[Epp95] D. Eppstein. Dynamic Euclidean minimum spanning trees and extrema of binary functions. *Discrete Comput. Geom.*, 13:111–122, 1995.

[GO97] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press LLC, Boca Raton, FL, 1997.

[HS94] D. Halperin and M. Sharir. New bounds for lower envelopes in three dimensions, with applications to visibility in terrains. *Discrete Comput. Geom.*, 12:313–326, 1994.