An Efficient Solution to the Zookeeper's Problem

John Hershberger
Mentor Graphics Corporation

Jack Snoeyink*
University of British Columbia

Abstract

Chin and Ntafos introduced the "Zookeeper's Problem" and gave a quadratic-time algorithm to solve it. We show how to improve their algorithm to $O(n \log^2 n)$.

1 Introduction

The Zookeeper's Problem is a shortest path problem introduced by Chin and Ntafos [1]. The problem is this: Given a simple polygon (the *zoo*) with a set of k disjoint convex polygons (the *cages*) inside it, each sharing one edge with the polygon, find a shortest cycle that starts at a given polygon vertex p (the *zookeeper's chair*), touches at least one point of each cage, and returns to p. See Figure 1. This path is the one the zookeeper should follow to feed all the animals while walking as short a distance as possible.

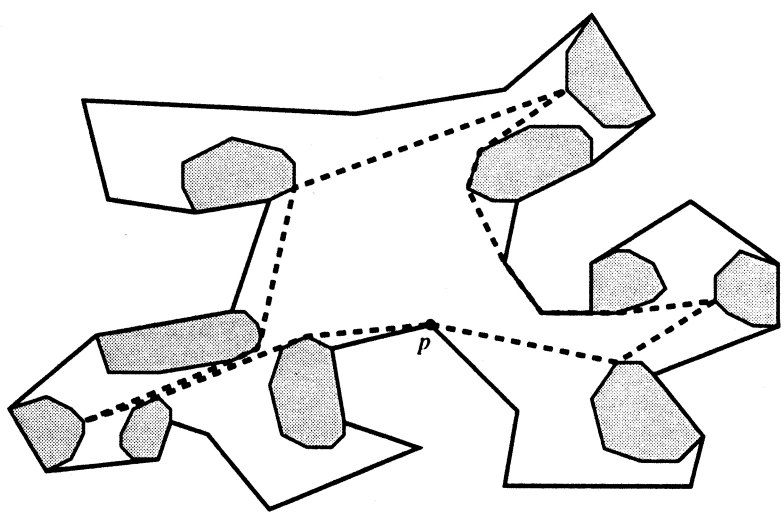

Figure 1: An instance of the zookeeper's problem

Clearly, the optimal path must visit the cages in the order they appear along the polygon boundary, since otherwise it would cross itself and could be shortened. Let P be the combined polygon consisting of all zoo and cage edges reachable from the zookeeper's chair,

*Supported in part by an NSERC Research Grant and a B.C. Advanced Systems Institute Fellowship.

and define $|P|$ to be n. Chin and Ntafos have shown how to find an optimum zookeeper's path in $O(n^2)$ time under the Real RAM machine model. In this note we show how to solve the problem in $O(n \log^2 n)$ time under the same model.

2 Background

The algorithm of Chin and Ntafos is based on the reflection principle: if a and b are two points on the same side of a line ℓ, the shortest path that visits a, ℓ, and b in that order can be computed by reflecting b across ℓ and drawing the straight line segment $\overline{ab'}$ from a to the reflection b' of b. The shortest path in the original setting is obtained by reflecting the part of $\overline{ab'}$ that connects ℓ to the reflection b', so that it connects ℓ to b. More generally, to find a shortest path from a to b that visits a line segment s, with a and b on the same side of the line defined by s, compute the shortest path that passes between the endpoints of s and connects a to the reflection b'. See Figure 2.

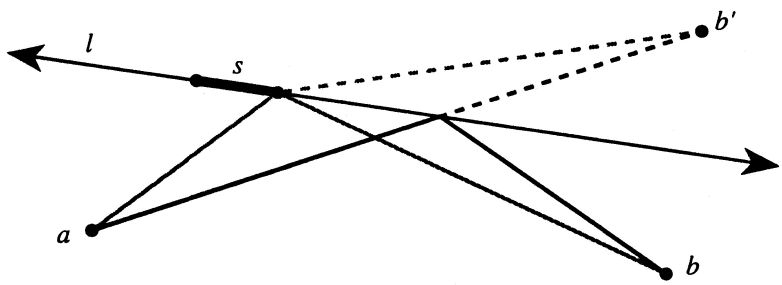

Figure 2: The reflection principle

The optimal zookeeper's path touches at least one segment of each cage. If we know some segment e_i that the optimal path touches on the ith cage (the optimal path may touch more than one), then we can use the reflection principle at each e_i to compute the path exactly. Let $\pi = (e_1, \ldots, e_k)$ be the combinatorial type of the path. Consider the topological "polygon" P_π we get by combining k copies of P, alternately mirror-reversed and normally oriented, with copy $i+1$ rotated and translated so that the image of e_i coincides with the image of e_i in copy i; both images of e_i are erased to connect the interiors of copies i and $i+1$. The resulting 2-manifold is simply connected; we can find shortest paths inside it by applying simple-polygon algorithms [4].

The observation of the previous paragraph reduces the zookeeper's problem to that of finding the combinatorial type of the optimal path. Chin and Ntafos do this by exploiting a monotonicity property of zookeeper's paths. They pick one combinatorial path type and modify it incrementally to get the optimal path type. They show that the length of the best path of the current type decreases monotonically, and that the type changes only $O(n)$ times before the optimal path type is found.

At a high level, the algorithm of Chin and Ntafos is as follows:

1. Orient the boundaries of odd-numbered cages clockwise and the boundaries of even-numbered cages counterclockwise; this gives each cage edge a head and a tail.

2. Initialize the path type π to be the one that includes the first edge of each cage (the one with no predecessors).

3. Compute the shortest path $path(\pi)$ from the zookeeper's chair to its image inside the topological polygon P_π.

4. **While** $path(\pi)$ passes through the head of some $e_i \in \pi$ that is not the last edge on cage i **do**

   (a) Replace e_i in π by its successor on the boundary of cage i.

   (b) Compute the shortest path $path(\pi)$.

Because each cage edge may be deleted from the path type at most once, there are $O(n)$ changes to the path type. Chin and Ntafos show that the length of $path(\pi)$ is non-increasing during this process, and that the process ends by finding the optimal path type. They show that each shortest path can be computed in $O(n)$ time, and so the process finds the optimal zookeeper's path in $O(n^2)$ time. The naïve algorithm would run in $O(n^3)$ time, since $|P_\pi| = \Theta(kn) = O(n^2)$.

3 Our contribution

We are able to compute each shortest path in $O(\log^2 n)$ time, which reduces the time of Chin and Ntafos's algorithm to $O(n \log^2 n)$. We use the shortest-path query structure of Guibas and Hershberger [2, 3]. In $O(n)$ time we preprocess the polygon (P, not the $O(n^2)$-size 2-manifold P_π) so that we can answer two-point shortest-path queries in $O(\log n)$ time. More particularly, we can compute an *hourglass* for any two edges of the polygon in $O(\log n)$ time. Updating the shortest path in P_π will involve computing at most $O(\log n)$ hourglasses.

3.1 Hourglasses

The hourglass defined by two polygon edges is the region of the polygon interior bounded by the shortest paths connecting the endpoints of the edges. See Figure 3.

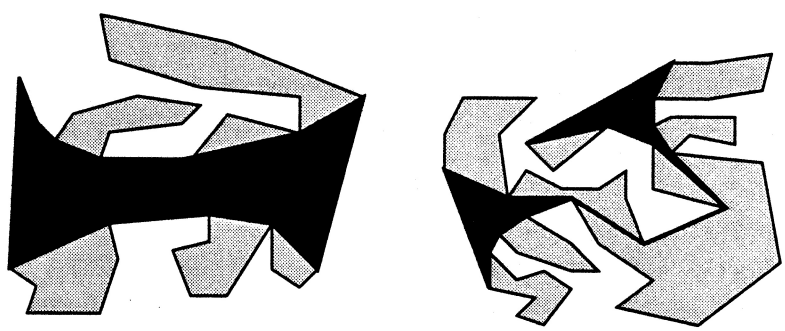

Figure 3: An open hourglass and a closed hourglass

Hourglasses are of two types, *open* and *closed*. An open hourglass is bounded by its two defining edges (say e and e') and by two convex chains that bulge toward the hourglass

interior. The two chains are arbitrarily called the *upper* and *lower* chains. There is at least one line that connects e to e' and separates the upper and lower chains. An hourglass is closed if there is no line segment that connects e to e' and lies completely inside the polygon. The interior of a closed hourglass has two components, a *funnel* $F(e,a)$ adjacent to e and a funnel $F(e',a')$ adjacent to e'. The funnel $F(e,a)$ is bounded by e and by the two convex shortest paths from a point a (the funnel *apex*) to the endpoints of e. $F(e',a')$ is defined similarly. The two funnels are joined in the closed hourglass by the shortest path connecting a to a' inside the polygon. (More generally, a funnel $F(e,p)$ is defined by the shortest paths from p to the endpoints of e; p need not be the funnel apex, i.e., the vertex where the shortest paths separate.)

An open hourglass is represented by a pair of binary trees, one for the upper chain and one for the lower chain. An hourglass is the concatenation of two sub-hourglasses. At the root of the tree for the upper chain is the common tangent between the upper chains of the two sub-hourglasses (the *bridge* edge); the left and right subtrees represent the upper chains of the sub-hourglasses. The structure of the tree for the lower chain is analogous. In a closed hourglass, the funnel chains are represented similarly.

An hourglass represents all shortest paths between the two edges. In the usual polygon setting in which hourglasses are used, one computes hourglasses between diagonals of the polygon, and *concatenates* hourglasses to make new hourglasses: given an hourglass between two diagonals a and b, and another between b and c, with diagonal b separating a from c, we can compute the hourglass between diagonals a and c in $O(\log n)$ time by computing a constant number of common tangents between the convex chains that form the hourglass boundary.

3.2 Reflected hourglasses

As mentioned before, the combinatorial type π of a path $path(\pi)$ is a sequence of cage edges $(e_1,\ldots,e_k)$, where e_i is on the ith cage. We denote the hourglass defined for edges e_i and e_{i+1} by $H_\pi(i,i+1)$. In order to extend this hourglass notation to the polygon P_π, which is obtained by reflecting, translating, and rotating the original polygon P, we augment the hourglass structure to store the necessary transformations.

Define $H_\pi(i,j)$ to be the hourglass that links a cage edge e_i to a cage edge e_j, applying the reflection principle at each cage edge e_h, for $i < h \leq j$. We represent $H_\pi(i,j)$ with e_i in its original position, and e_j reflected, rotated, and translated to some image position. We store with the hourglass the homogeneous coordinate transformation matrix needed to move e_j from its natural position to its image's position in $H_\pi(i,j)$, including the reflection at e_j. When we concatenate two hourglasses $H_\pi(i,h)$ and $H_\pi(h,j)$, we transform coordinates from the second hourglass $H_\pi(h,j)$ into the frame of the first using the transformation matrix stored with the first. (For hourglasses that are properly contained in some $H_\pi(i,i+1)$, the identity matrix is stored.)

To concatenate two hourglasses, we must compute common tangents between pairs of convex chains, one chain in each hourglass. Computing common tangents requires us to make tests based on the coordinates of triples of points. The points we use are the endpoints of the bridge edges stored at the subtree roots of each hourglass. We compute the coordinates of these points in the frame of reference of the hourglass that we are trying

to construct. The transformation matrix needed to convert a sub-hourglass's coordinates into the top-level frame of reference is the product of a sequence of transformation matrices. Each hourglass node has a transformation matrix associated with it; the product sequence for a node v is obtained by taking, in top-to-bottom order, the matrices associated with left children of the path from the root to v. Because the points we test belong to a descending sequence of tree nodes, we can maintain a current transformation matrix at each step. Whenever we descend into the right subtree of the current node v, we update the current matrix by multiplying it by the matrix of v's left child.

3.3 Paths as degenerate hourglasses

The optimal zookeeper path is equivalent, via the reflection principle, to a shortest path that starts from p, goes through the hourglass $H_\pi(1,k)$, and ends at the image of p transformed by the matrix associated with $H_\pi(1,k)$. Once the hourglass $H_\pi(1,k)$ and its transformation matrix are known, the image of the shortest path can be found in $O(\log n)$ time as follows: compute the funnel $F(e_1,p)$ from p to e_1 and the funnel $F(e_k,p)$ from e_k to p; transform the latter funnel using the matrix of $H_\pi(1,k)$; concatenate $F(e_1,p)$, $H_\pi(1,k)$, and the transformed image of $F(e_k,p)$. This produces the shortest path $path(\pi)$ as a degenerate "closed hourglass" from p to p. By abuse of notation, we could call this degenerate hourglass $H_\pi(0,k+1)$. There are at most $O(n)$ edges in $path(\pi) = H_\pi(0,k+1)$, since the zookeeper path turns only at edges and vertices of P and visits each edge or vertex at most once. The edges of $path(\pi)$ can be produced in $O(n)$ time [2, 3]. We show how to compute $path(\pi)$ in $O(n \log n)$ time, how to use $path(\pi)$ to find where π should change, and how to update $path(\pi)$ in $O(\log^2 n)$ time when π changes.

We build $path(\pi) = H_\pi(0,k+1)$ by bottom-up merging according to a complete binary tree with the hourglasses $H_\pi(i,i+1)$ at the leaves. The transformation matrix $T(i,i+1)$ associated with $H_\pi(i,i+1)$ is the matrix that reflects the plane across the line supporting e_{i+1}. Whenever we concatenate two hourglasses $H_\pi(i,h)$ and $H_\pi(h,j)$, we multiply their transformation matrices. Each concatenation takes $O(\log n)$ time, and there are $O(n)$ of them, so the total time to build the initial hourglass is $O(n \log n)$.

The edges e_i and e_{i+1} are directed oppositely on the boundary of $H_\pi(i,i+1)$, one clockwise and one counterclockwise. Let us define the upper chain of hourglass $H_\pi(i,i+1)$ to be the chain incident to the heads of the two edges, and the lower chain to be the one incident to the tails. Observe that the transformation matrices ensure that all the upper chains lie on one side of $H_\pi(1,k)$ and all the lower chains lie on the other side. Any edge e_i whose head lies on $path(\pi)$ is an edge that can be replaced in the path type π by its successor e_i'. For each hourglass, we maintain a list of edges e_i whose heads are on the upper hourglass chains (these may belong to $path(\pi)$), and another of edges whose heads are on the collapsed part of a closed hourglass (these definitely will belong to $path(\pi)$). When we concatenate two hourglasses, it is not difficult to compute the lists for the resulting hourglass in the same $O(\log n)$ time required for the rest of the concatenation operation (we represent the lists with balanced binary trees). The shortest path $path(\pi)$ is just a degenerate hourglass, and we can select from its "definite" list an edge that should be replaced in π by its successor. If an edge e_i has its head on the final shortest path, but it is the last edge of its cage, then we cannot replace it by its successor. In this case we remove e_i from all the lists, so it will

no longer be considered for replacement.

When the path type π changes to some π', it is because one of the edges e_i has been replaced by its successor e_i'. The hourglasses in the merge tree that need to be updated are those that are incident to e_i or contain it in their interior—namely, the two hourglasses $H_\pi(i-1,i)$ and $H_\pi(i,i+1)$, plus their $O(\log n)$ ancestors in the merge tree. By merging bottom-up, we transform the hourglass $H_\pi(0,k+1) = path(\pi)$ into $H_{\pi'}(0,k+1) = path(\pi')$ in $O(\log^2 n)$ time.

4 Conclusion

We have shown how to reduce the time needed for the inner loop of the zookeeper algorithm to $O(\log^2 n)$. As Chin and Ntafos show, there are only $O(n)$ executions of the inner loop, so our data structure reduces the running time of the zookeeper algorithm to $O(n \log^2 n)$.

Remark: Zookeeper algorithms based on the reflection principle may be far from practical. They are stated using the Real RAM model of computation, which is standard for computational geometry algorithms, but that model seems particularly inappropriate. These algorithms require the multiplication of $O(n)$ transformation matrices. If the input coordinates have L bits of precision, the transformation matrices will have $O(nL)$ bits. If the computation is performed using finite-precision floating point, the numerical inaccuracy inherent in the computation is likely to cause errors in the combinatorial path structure in all but the smallest examples.

References

[1] W. Chin and S. Ntafos. Optimum zoo-keeper routes. *Congressus Numerantium*, 1989.

[2] L. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.

[3] J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38:231–235, 1991.

[4] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. In *Proceedings of the 2nd Workshop on Algorithms and Data Structures*, pages 331–342. Springer-Verlag, 1991. Lecture Notes in Computer Science 519.