Algorithms for the Minimum Diameter of Moving Points and for the Discrete 1-center Problem

Kenneth L. Clarkson
Bell Laboratories, Lucent Technologies
Murray Hill, New Jersey 07974
clarkson@research.bell-labs.com
http://cm.bell-labs.com/who/clarkson/

March 1997

Abstract

Given points moving with constant, but possibly different, velocities, the minimum moving diameter problem is to find the minimum, over all time, of the maximum distance between a pair of points at each moment. This note gives a randomized algorithm requiring $O(n \log n)$ expected time for this problem, in two and three dimensions. Also briefly noted is a randomized $O(n \log n \log \log n)$ expected-time algorithm for the discrete 1-center problem in three dimensions; in this problem, a member p of a set S of points is desired, whose maximum distance to S is minimum over all points of S.

1 Introduction

Given a set S of n sites (points), the *discrete 1-center problem* is to find the point of S minimizing the maximum distance to points of S. That is, for point p, let $d_{\max}(p, S)$ denote $\max_{q \in S} d(p, q)$, where $d(p, q)$ is the Euclidean distance between p and q. The discrete 1-center problem is to find $\min_{p \in S} d_{\max}(p, S)$, and the point p that realizes that distance.

The input for the *minimum moving diameter problem* is a set S of sites that are each moving with constant, but possibly different, velocities: that is, at time t, the position of site p is a point $p(t) = m_p + tv_p$, where m_p is the position of site p at time $t = 0$, and v_p is the velocity of site p. The sites give rise to a collection of points $S(t)$ at time t, their position at that time. The minimum moving diameter problem is to find

$$\min_t \max_{p \in S(t)} d_{\max}(p(t), S(t)).$$

This note gives randomized algorithms for these problems; the moving diameter algorithm requires $O(n \log n)$ expected time, in two and three dimensions.

Previously Gupta *et al.*[5] gave an algorithm needing $O(n \log^3 n)$ worst-case time in the plane. Their algorithm, using parametric search, has been implemented;[] it may be that the algorithm here is easier to code.

The algorithm given here for the discrete 1-center problem in three dimensions needs $O(n \log n \log \log n)$ expected time. An algorithm for the discrete 1-center problem in the plane needing $O(n \log n)$ time is easy to find using the farthest point Voronoi diagram, but the author has found no previous work on the three-dimensional version of the problem.

Both algorithms will use an algorithm for the intersection of equal-radius balls in three dimensions;[3] such intersections have a descriptive complexity linear in the number of input balls. The use of ball intersections is not surprising here, since the prior use of that algorithm was to help find the diameter of a point set, $\max_{p \in S} d_{\max}(p, S)$. Algorithms for planar point location can be adapted to determine the location of a given point with respect to a ball intersection, that is, whethere the point is inside or outside the intersection.[3] The usefulness of that location algorithm here is that, if a point is inside the intersection, then it is closer than ρ to each of the ball centers. Here ρ is the radius of the balls.

The minimum moving diameter problem is quite easily expressed as a convex programming problem, and the algorithm applies ideas previously used for linear programming and for convex programming.[3, 1] Curiously, the most natural statement of the problem in terms of convex programming has $\Omega(n^2)$ constraints; the most obvious approach then requires $\Omega(n^2)$ time. However, as will be apparent, not all $\Omega(n^2)$ constraints need be explicitly considered, and so the algorithm can be more efficient.

2 The minimum moving diameter algorithm

As Gupta *et al.* observe, the squared distance between two given sites is a quadratic function of time; in other words, the graph of $d(p(t), q(t))^2$ vs. t is a parabola, where $p(t)$ and $q(t)$ denote the position of the moving sites at time t. Each pair of points gives a different parabola, and at time $\hat{t}$, the diameter is the maximum ordinate of the intersections of these parabolas with the line $t = \hat{t}$. That is, the diameter at different times gives the upper envelope of the parabolas, or equivalently, the boundary of the intersection of the regions above these parabolas. Thus the min-diameter problem is to find the point (t^*, y^*) in that intersection of parabolic regions with y^* as small as possible. Since each parabolic region is convex, this is simply a convex programming problem in two dimensions, with a set $\mathcal{P}$ of $\binom{n}{2}$ constraints: the constraints are the parabolic regions, and the "objective function" value at a point is its ordinate. Using the techniques of Adler and Shamir[1], generalizing from a linear programming algorithm[2], these problems can be solved, for S in any dimension, in time linear in the number of constraints, that is, in $O(n^2)$ time. This is a nontrivial, albeit obvious, result in more than three dimensions.

The faster algorithm is a variation on this old algorithm; after a few remarks, we'll briefly describe the old algorithm, and then the changes that speed it up.

For technical reasons, we note that the optimum point for this convex programming problem is unique, or can be viewed as unique, using the convention that if the minimum diameter is realized over of period of time, rather than at an instant, we use as optimum point the one with smallest t value realizing the minimum diameter. If there is no such smallest t value, we view the optimum as occurring at $t = -\infty$. This uniqueness is needed for technical reasons.

The convex programming optimum here is determined by a set $\mathcal{P}^*$ of two constraints; that is, the optimum for the two constraints in $\mathcal{P}^*$, by themselves, is the optimum for all of $\mathcal{P}$. The goal is thus to find that subset $\mathcal{P}^*$. The algorithm will be applied recursively, to some subset of the original set of constraints $\mathcal{P}$.

The old algorithm takes a random subset R of the constraints, and recursively computes the optimum for that subset. It then finds the set $\mathcal{V}$ of constraints violated by that optimum. If $\mathcal{V}$ does not contain a member of $\mathcal{P}^*$, then an optimum has been found: if both members of $\mathcal{P}^*$ are unviolated, then the objective function value cannot be any better than the optimum y^*; on the other hand, that value cannot be any worse than y^*, since fewer constraints are satisfied. Therefore, the optimum found for R has the same objective function value as the optimum for $\mathcal{P}$, and is therefore the unique optimum.

In short, if $\mathcal{V}$ is empty, the algorithm stops, and otherwise $\mathcal{V}$ contains a member of $\mathcal{P}^*$. Here the algorithm recursively computes the optimum for $R' \cup \mathcal{V}$, where R' is another random subset of $\mathcal{P}$. Let $\mathcal{V}'$ be the set of constraints violated by the optimum for $R' \cup \mathcal{V}$. The set $\mathcal{V}'$ is either empty, and the algorithm is done, or $\mathcal{V}'$ contains the remaining member of $\mathcal{P}^*$, and the optimum for $\mathcal{V} \cup \mathcal{V}'$ is optimum overall. The recursive computation of the optimum for $\mathcal{V} \cup \mathcal{V}'$ completes the algorithm.

To speed up the algorithm, we do not use the sets of constraints $\mathcal{V}$ and $\mathcal{V}'$, but rather implicit representations for supersets of these constraints. Specifically, we represent $\mathcal{V}$ by the set of sites $S_\mathcal{V}$ that gave rise to $\mathcal{V}$: each constraint in $\mathcal{V}$ yields two elements of $S_\mathcal{V}$, the two sites associated with that constraint. Thus $|S_\mathcal{V}| \leq 2|\mathcal{V}|$. We represent $\mathcal{V}'$ similarly, as $S_{\mathcal{V}'}$. Let $\mathcal{P}(S_\mathcal{V})$ denote the set of constraints associated with any two members of $S_\mathcal{V}$. Clearly $\mathcal{V} \subset \mathcal{P}(S_\mathcal{V})$.

With this in mind, in the general step, the algorithm will be called with a set of constraints represented as the union of two sets X and $\mathcal{P}(M)$, where set X is a set of explicit constraints, and $\mathcal{P}(M)$ is the set of implicit constraints, associated with a set of sites $M \subset S$. Initially X is empty and $M = S$.

It is easy to obtain a random subset of this set of constraints without explicitly examining the constraints in $\mathcal{P}(M)$; for example, consider the constraints in a canonical numbering $1 \ldots \binom{m}{2}$, where m is the size of M. Pick random numbers in that range repeatedly, tossing out duplicates, until a set equal to the desired random subset size is found. By inverting the canonical numbering, the constraints are found. Since the expected size of the set is at least $s - s(s-1)/2\binom{m}{}$ after s trials, and $s < K_m m$, the expected value of s is $m + O(1)$; using balanced trees to maintain the current subset and check for duplicates, the expected work is $O(s \log m) = O(m \log m)$.

The size of R is $r \equiv \max\{K_x \sqrt{x}, K_m m\}$, where x is the size of the explicit set X, m is the size of the set of sites M for the implicit set, and K_x and K_m

are sufficiently large constants. The total number of constraints represented is $x + \binom{m}{2}$, since each pair of sites in M implies a constraint.

Given a subset optimum (t', y'), the explicit constraints X_V violated by that optimum can be found readily; the violated implicit constraints can be found as follows. A constraint is violated by (t', y') if and only if the intersection of the constraint's boundary with the vertical line $t = t'$ is a point (t', y'') with $y'' > y'$; the value y'' is simply the squared distance, at time t', between the two sites p and q associated with the constraint. In short, the violated constraints correspond to sites M_V whose squared distances at time t' are larger than y'. Thus the violated constraints of $\mathcal{P}(M)$ are contained in the constraints of $\mathcal{P}(M_V)$.

Such sites M_V can be found using the same methods as in an algorithm for computing the diameter of a point set[3]: given radius ρ (in the above, $\sqrt{y'}$), compute the intersection of the balls of radius ρ centered at the sites, as positioned at time $t = t'$. Find all sites outside that intersection; these sites have farthest point distance greater than ρ at time t'.

Now a second random subset R' of the constraints is found, again of size r, and the algorithm is called recursively with the explicit constraint set $R' \cup X_V$, and the implicit constraint set represented by M_V. The result is another pair of sets $X_{V'}$ and $M_{V'}$, representing violated constraints of X and $\mathcal{P}(M)$. The algorithm is called recursively again with explicit set $X_V \cup X_{V'}$ and implicit set $M_V \cup M_{V'}$, and the result is the overall optimum.

The recursion stops when x and m are smaller than a constant K, at which point some algorithm for convex programming is called for such constant-size problems.

For convenience of analysis, the algorithm is changed as follows: rather than a single recursive call with random subset R, the algorithm repeats with different random subsets until M_V and X_V are below an appropriate threshold in size. This threshold is $6m^2/r$ for M_V and $6x/r$ for X_V. The same repetition is done for finding $M_{V'}$ and $X_{V'}$.

Theorem 1 *The minimum diameter of a moving point set in two or three dimensions can be found by a randomized algorithm in expected $O(n \log n)$ time.*

Proof. To solve a given problem, the algorithm finds three optima recursively. Using the first two optima, the algorithm does $O(x + m \log m)$ expected work overall to find the sets M_V, X_V, $M_{V'}$, and $X_{V'}$.

The expected sizes of X_V and $X_{V'}$ are $2x/r$, and the expected sizes of M_V and $M_{V'}$ are at most $4\binom{m}{2}/r < 2m^2/r$. (Perhaps first explicitly claimed by Adler and Shamir[1], such bounds are a ready consequence of random sampling results,[3], and can also be shown using "backwards analysis."[7, 4]) By Markov's inequality, the probability that one of these sets exceeds 3 times its expected size is $1/3$; hence an iteration to find a random subset R with these sizes small will succeed with probability $1/3$. Let $T(x, m)$ denote the expected work for a problem with x explicit constraints and m sites for implicit constraints. The total work for finding a subset R giving small X_V and M_V is

$$T(r,0) + \frac{2}{3}T(r,0) + \frac{4}{9}T(r,0)\ldots,$$

or expected $3T(r,0)$. Similarly, the expected work to find small $M_{V'}$ and $X_{V'}$ is $3T(r+6x/r, 6m^2/r)$. Finally, the work to find the optimum for the final recursive call is $T(12x/r, 12m^2/r)$. Therefore, the expected work $T(x,m)$ is bounded by

$$T(x,m) \leq O(x)+O(m\log m)+3T(r,0)+3T(r+6x/r,6m^2/r)+T(12x/r,12m^2/r),$$

for x or m larger than K, and $O(1)$ for small x and m. It is easy to obtain the bound $T(x,m) = O(x+m\log m)$ from this recurrence, for $K_x = 6\sqrt{5}$, $K_m > 30$, and $K \geq 5$. The theorem follows. □

3 The three-dimensional discrete 1-center problem

Given a set S of n sites (points), the discrete 1-center problem is to find the site c whose maximum distance to the remaining sites is minimal, over all sites. That is, the maximum distance to c is

$$\min_{p\in S} d_{\max}(p,S),$$

where $d_{\max}(p,S) \equiv \max_{q\in S} d(p,q)$, and $d(p,q)$ is the Euclidean distance from p to q. We can generalize this problem slightly: given sets T and S, let

$$c(T,S) \equiv \min_{p\in T} d_{\max}(p,S),$$

so that $c(S,S)$ is the discrete 1-center distance of S.

The problem can be solved recursively as follows: in the general step, at recursion depth k, if k at least $\lg\lg n$, use a simple algorithm for $c(T,S)$ requiring $O(n|T|)$ time. Otherwise, pick a random subset R of T, of size $r_k = \sqrt{m_k}$, where $m_k \equiv m^{2^{-k}}$, and m is the size of the top level set T. Recursively compute $c(R,S)$, giving a distance $\rho = c(R,S)$. For each site $q \in S$, consider the ball with radius ρ centered at q. All sites of T closer than ρ to q are inside that ball. If a site $p \in T$ is in all such balls, over all $q \in S$, then its maximum distance to the other sites is no more than ρ, so $d_{\max}(p,S) \leq \rho$. The set T' of such sites inside the ball intersection for S are the only possible points of T with maximum distance to S equal to $c(T,S)$. The set T' can be found in $O(n\log n)$ time, or even in $O(n\log m)$ expected time using the slightly more complicated algorithm described below. Now compute $c(T',S)$, which is $c(T,S)$, and the algorithm is done.

It remains to describe how to find the set $T' \subset T$ inside all balls of radius ρ centered at sites of S. This can be done as follows. Split the set S arbitrarily into

$\lceil n/m \rceil$ groups, and for each group, compute ball intersections and use planar point location to test each member of T for inclusion in the ball intersection. A member of T' must be in all such ball intersections.

Theorem 2 *Given sets T and S in three dimensions, the value of $c(T, S)$ can be found in $O(n \log m \log\log m)$ expected time, The discrete 1-center of S can be found in $O(n \log n \log\log n)$ expected time.*

Proof. The expected size of T at recursion depth k is m_k; this follows inductively from the claim that the expected size of T' is $|T|/r_k$. To prove the latter, pick $p \in T$ at random, and consider $c(R \cup \{p\}, S)$. Since p is a random element of $R \cup \{p\}$, the probability that $c(R \cup \{p\}, S) < c(R, S)$ is $1/(r_k + 1)$: these are different only if p is the element of $R \cup \{p\}$ determining $c(R \cup \{p\}, T)$. But this condition is the same as $c(p, S) = c(R \cup \{p\}, S)$. Hence a random member of T has probability $1/(r_k + 1)$ of having smaller max-distance to S than $c(R, S)$, and the expected size of T' is $|T|/(r_k + 1)$.

The work to find and test the ball intersections is $O(n \log |T|)$, since each group from S requires $(|T| \log |T|)$ time for the ball intersections and inclusion testing, and so all such testing needs $O(n \log |T|)$ time. From the concavity of the logarithm function, the expected work to make and test the ball intersections is $O(n \log m_k)$, at recursion depth k.

The expected work W_k at depth k satisfies

$$W_k \leq 2W_{k+1} + O(n \log m_k),$$

for $k < \lg\lg n$, and $O(m_k n)$ for $k \geq \lg\lg n$. This has the bound

$$W_k \leq \lceil (\lg\lg m_k) - k \rceil n \log m_k,$$

which implies the time bound for finding $c(T, S)$. The bound for the discrete 1-center problem follows trivially. □

4 Concluding remarks

Megiddo's parametric search technique has proven very useful in computational geometry; sometimes randomization gives an alternative approach that is simpler or faster, although with only expected bounds. This note is another example of this phenomenon.

A curious feature of the minimum moving diameter algorithm is that it allows a convex programming problem with n^2 constraints to be solved in $o(n^2)$ time, because many unviolated constraints can be ignored without considering them explicitly. Can this trick work elsewhere?

When the recursive approach used here was introduced for linear programming, another approach using iterative reweighting was also given.[2] Applying that alternative to the minimum moving diameter problem doesn't seem to yield an algorithm with quite as small a running time: the farthest point queries are always done to the whole set S. The same apparently holds also for later algorithms,[7, 6] which are more "complete" in the sense that their base case problem has very few constraints. The recursive technique gives a batch of queries to be answered at once. This is not unlike parametric search, where a parallel algorithm gives a batch of a certain kind of queries, that can be answered quickly.

The time bound given here for the discrete 1-center problem seems unnatural. Is a bound of $O(n \log n)$ possible?

References

[1] I. Adler and R. Shamir. A randomization scheme for speeding up algorithms for linear and convex quadratic programming problems with a high constraints-to-variables ratio. *Math. Prog.*, 61:39–52, 1993.

[2] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM*, 42(2):488–499, 1995. Preliminary publicaton in *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, 1988.

[3] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[4] Bernd Gärtner and Emo Welzl. Linear programming – randomization and abstract frameworks. In *Proc. 13th Sympos. Theoret. Aspects Comput. Sci.*, volume 1046 of *Lecture Notes Comput. Sci.*, pages 669–687. Springer-Verlag, 1996.

[5] P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. *Comp. Geom.: Theory and Appl.*, 6(6):371–392, 1996.

[6] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 1–8, 1992.

[7] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete & Computational Geometry*, pages 423–433, 1991.