THE REGION APPROACH FOR SOME DYNAMIC CLOSEST-POINT PROBLEMS.

EXTENDED ABSTRACT

June 27, 1994

ABSTRACT.

Let S be a set of n points in the space $\mathbb{R}^k$ under the L_t metric. We consider the following dynamic problems

1) finding a nearest neighbor in S of any query point for $t = 1, \infty$.
2) finding a $(1+\varepsilon)$-nearest neighbor in S of any query point for $t \in]1, \infty[$.
3) maintenance of a closest pair of S.

Applying the region approach we reduce these problems to dynamic problem of range searching for maximum.

Chazelle's data structure for range searching for maximum [2] allows to achieve query and update times of $O(\log^{k+1} n \log\log n)$, using $O(n \log^{k-2} n)$ space. Previously, no linear size data structure having polylogarithmic update time was known for maintenance of a closest pair of S in $\mathbb{R}^2$.

1. INTRODUCTION

Proximity problems in computational geometry are well studied. In this paper we consider the dynamic version of the nearest neighbor problem and the closest pair problem. We are given a set S of n points in k-dimensional space $\mathbb{R}^k$. The set S is changed by insertions and deletions of points. In the closest pair problem we have to compute a closest pair of S after each update. Distances are measured in the Minkowski L_t-metric, where $1 \leq t \leq \infty$. In the nearest neighbor problem we have to compute a point in S nearest to a query point.

Gabow, Bentley and Tarjan [3] reduced the $L_\infty(L_1)$-neighbor problem to orthant searching for minimum. In $\mathbb{R}^k_1$ ($\mathbb{R}^k_\infty$), $k \geq 1$ they achieved $O(\log^{\max(k-1,1)} n)$ query time,
$O(n \log^{\max(k-1,1)} n)$ preprocessing time and $O(n \log^{k-1} n)$ space. For the static version of the L_∞- neighbor problem, S. Kapoor and M. Smid [5] gave a data structure of size $O(n \log^{k-2} n)$ that finds an L_∞-neighbor of query point in $O(\log^{k-1} n)$ time. They solved the dynamic version of the L_∞-neighbor problem with a query time of $O(\log^{k-1} n \log\log n)$ and an amortized update time of $O(\log^{k-1} n \log\log n)$, using $O(n \log^{k-1} n)$ space.

In the approximate neighbor problem we have to compute a $(1+\varepsilon)$-approximate neighbor of a query point (the distance to $(1+\varepsilon)$-approximate neighbor is at most

---

Key words and phrases. region approach, dynamic maintenance, data structures.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

$(1+\varepsilon)$ times a distance to the closest neighbor). S. Kapoor and M. Smid [5] presented a data structure with a query time of $O(\log^{k-1} n \log\log n)$ and an amortized update time of $O(\log^{k-1} n \log\log n)$, using $O(\frac{1}{\varepsilon^{k-1}} n \log^{k-1} n)$ space. The algorithm uses Yao's construction of regions [14].

There are several algorithms for the dynamic closest pair problem [5,6,8,9,10,11,12]. C. Schwarz gives [9] a survey of the dynamic closest pair algorithms. In [6,8,10] the problem is solved with $O(\sqrt{n}\log n)$ update time using $O(n)$ space. S. Kapoor and M. Smid [5] gave a data structures of size $S(n)$ which maintain the closest pair in $U(n)$ amortized time per update, where for $k \geq 3$, $S(n) = O(n)$ and $U(n) = O(\log^{k-1} n \log\log n)$; for $k = 2$, $S(n) = O(n \log n/(\log\log n)^m)$ and $U(n) = O(\log n \log\log n)$; for $k = 2$, $S(n) = O(n)$ and $U(n) = O(\log^2 n/(\log\log n)^m)$ (m is an arbitrary non-negative integer constant).

In Section 2 we briefly describe the orthant-based approach to $\mathbb{R}_1^k$ and $\mathbb{R}_\infty^k$ post office problem of Gabow, Bentley and Tarjan [3]. Also we show how to reduce the number of regions in $\mathbb{R}_\infty^k$. In Section 3 we give the explicit construction of regions and reduce the $(1+\varepsilon)$-approximate neighbor problem to the dynamic problem of range searching for maximum. In Section 4 we use the region construction from Section 3 and reduce the dynamic closest pair problem to the dynamic problem of range searching for maximum. The update algorithms (except the range searching) are simple.

Chazelle's data structure for range searching for maximum [2] allows to achieve query and update times of $O(\log^{k+1} n \log\log n)$, using $O(n \log^{k-2} n)$ space. Previously, no linear size data structure having polylogarithmic update time was known for maintenance of a closest pair of S in $\mathbb{R}^2$.

2. The nearest neighbor problem for L_1, L_∞

Gabow, Bentley and Tarjan [3] reduced the post office problem in $\mathbb{R}_1^k$ ($\mathbb{R}_\infty^k$) to orthant searching for minimum. We use this reduction in the dynamic version of problem.

For L_1-metric, the regions are the orthants. The number of regions is 2^k. Let q be a query point and $q_1, \ldots, q_{2^k}$ are the region neighbors of q in the orthants. A nearest neighbor of q is one of the points $q_1, \ldots, q_{2^k}$. We use 2^k data structures for the region neighbor problems. To find a nearest neighbor in S of a query point q we

1) find 2^k region neighbors of q and

2) choose a nearest neighbor which has minimal distance to q.

The insertion (resp. deletion) algorithm insert (resp. delete) a point into (resp. from) 2^k data structures.

Consider one of the regions at q. This region has form $R = \{x : a_i(x_i - q_i) \geq 0, a_i \in \{\pm 1\}\}$. For a point p in R, the distance between p and q is $\delta(p) - \delta(q)$. $\delta(x)$ is a *derived distance function* $\delta(x) = \sum a_i x_i$. This allows a region neighbor to be chosen as a point that minimizes δ. A region neighbor of q maximizes function $-\delta$. Therefore we reduced the dynamic problem of finding a region neighbor in S to the dynamic problem of range searching for maximum. Several data structures are proposed for the dynamic problem of range searching in [2,7,13]. Chazelle's data structure for range searching for maximum [2] allows to achieve query and update times of $O(\log^{k+1} n \log\log n)$, using $O(n \log^{k-2} n)$ space.

For L_∞-metric Gabow, Bentley and Tarjan [3] proposed $2^k k!$ regions which cover the space. For any region, the distances between the center of region (i.e. the origin) and a point of the region is measured by a derived distance function. This means that the intersection of the region and the unit sphere (i.e. k-cube with side 2) lies in some face of the k-cube. In fact the condition of narrowness is no necessary. We can partition each face of the unit sphere into $(k-1)!$ simplices [4]. Each simplex corresponds to some region. Hence we obtain $2k \cdot (k-1)! = 2k!$ regions. Furthermore we can use the connection between the regions and the triangulation of the $(k-1)$-cube [1]. This gives $2k\tau_{k-1}$ regions where τ_{k-1} is the minimum number of simplices to triangulate the $(k-1)$-cube. For $k = 2, 3, 4, 5$ the number of regions is $4, 12, 40, 160$ respectively.

2.1 Theorem. *Let some algorithm solve the dynamic problem of range searching for maximum in $U(k,n)$ update $Q(k,n)$ query and $P(k,n)$ preprocessing time, using $S(k,n)$ space. The dynamic problem of finding $L_1(L_\infty)$-neighbor can be solved in $2^k(U(k,n)+O(k\log n))$ (resp. $2k\tau_{k-1}(U(k,n)+O(k\log n))$) update $2^k(Q(k,n)+O(k\log n))$ (resp. $2k\tau_{k-1}(Q(k,n)+O(k\log n))$) query and $2^k P(k,n)$ (resp. $2k\tau_{k-1}P(k,n)$) preprocessing time, using $2^k S(k,n)$ (resp. $2k\tau_{k-1}S(k,n)$) space.*

3. The approximate L_t-neighbor problem

This Section applies the region approach to the approximate L_t-neighbor problem. Let S be a set of n points in $\mathbb{R}^k$ and let $\varepsilon > 0$ be a fixed constant. For any point $p \in \mathbb{R}^k$, a point $q \in S$ is a $(1+\varepsilon)$-approximate L_t-neighbor of p if $d_t(p,q) \le (1+\varepsilon)\min\{d_t(p,r) : r \in S\}$. We shall give an explicit construction of regions that allows to reduce the $(1+\varepsilon)$-approximate L_t-neighbor problem to the range searching for maximum.

The main idea is that instead of L_t-distances we use a distances which measured by a derived distance function. This function is dependent on a region. A set of regions is dependent on ε.

Let $B = \{b_1, \ldots, b_k\}$ be a basis of $\mathbb{R}^k$. For any $p \in \mathbb{R}^k$, the *region of B at p* is defined as

$$R(B,p) = \{p + \sum_{i=1}^{k} \lambda_i b_i : \lambda_i \ge 0, b_i \in B\}.$$

For this region we define a derived distance function δ such that, for a point $x = \sum_{i=1}^k \lambda_i b_i \in \mathbb{R}^k$, $\delta(x) = \sum_{i=1}^k \lambda_i \|b_i\|$. For a point $x = p + \sum_{i=1}^k \lambda_i b_i \in R(B,p)$,

$$\delta(x) = \delta(p + \sum_{i=1}^{k} \lambda_i b_i) = \delta(p) + \sum_{i=1}^{k} \lambda_i \|b_i\| \ge \delta(p) + d_t(p,x).$$

Instead of the distance $d_t(p,x)$ we use $\delta(x) - \delta(p)$. We call a region of B at p a $(1+\varepsilon)$-*narrow* region if, for any point $x \in R(B,p)$

$$d_t(p,x) \le \delta(x) - \delta(p) \le (1+\varepsilon) d_t(p,x).$$

Let $R(B,0)$ be a $(1+\varepsilon)$-narrow region. For a query point p, find a point $q \in S \cap R(B,p)$ that minimizes δ. q is a $(1+\varepsilon)$-approximate region neighbor of p. A $(1+\varepsilon)$-approximate neighbor of p can be chosen among $(1+\varepsilon)$-approximate region neighbors of p for regions which cover the space.

3.1 Theorem. *For any $\varepsilon > 0$, there exists a family of $(1+\varepsilon)$-narrow regions at origin such that the regions cover the space and the number of regions is $O(\frac{1}{\varepsilon^{k-1}})$.*

For a point x, let x' denote $\frac{x}{\|x\|}$.

3.2 Lemma. *Let $s = \langle s_1, \ldots, s_k \rangle$ be a simplex in $\mathbb{R}^k_t$ such that $\|s_i\| \geq 1$ and $\|s_i - s_j\| \leq \varepsilon/4$ where $\varepsilon \in (0,1)$. Then the region corresponding to the simplex s is $(1+\varepsilon)$-narrow.*

3.3 Lemma. *Let a, b are a points in $\mathbb{R}^k_t$ and $\|a\| \geq 1$, $\|b\| \geq 1$, $\|a-b\| \leq \varepsilon/2$ where $\varepsilon \in (0,1)$. Then $\|a' - b'\| \leq \varepsilon$.*

3.4 Theorem. *Let some algorithm solve the dynamic problem of range searching for maximum in $U(k,n)$ update $Q(k,n)$ query and $P(k,n)$ preprocessing time, using $S(k,n)$ space. The $(1+\varepsilon)$-approximate L_t-neighbor problem can be solved in $\frac{c(k)}{\varepsilon^{k-1}}(U(k,n) + O(k\log n))$ update $\frac{c(k)}{\varepsilon^{k-1}}(Q(k,n) + O(k\log n))$ query and $\frac{c(k)}{\varepsilon^{k-1}}(P(k,n) + O(n\log n))$ preprocessing time, using $\frac{c(k)}{\varepsilon^{k-1}}S(k,n)$ space. $c(k)$ depends on the dimension k only.*

Chazelle's data structure for range searching for maximum [2] allows to achieve query and update times of $O(\log^{k+1} n \log\log n)$, using $O(n\log^{k-2} n)$ space.

4. The maintenance of a closest pair

This Section explores the region approach for the closest pair problem. A point $p \in S$ is a *nearest neighbor of* q if, for any $r \in S$, $d_t(p,q) \leq d_t(q,r)$. For a points $p, q \in S$, we call the pair (p,q) a *neighbor pair* if p is a nearest neighbor of q and vice versa. It is clear that the closest pair of S is a neighbor pair of S. Instead of L_t-distances we use a distances which measured by a derived distance function. The derived distance function is dependent on a region. We shall construct a finite family of regions at common center (the origin). The regions cover the space $\mathbb{R}^k$.

Let p be a point in $\mathbb{R}^k_t$, R be a region at p, δ be a derived distance function, and q be a point in $S \cap R$ that minimizes δ. We call q a *δ-region neighbor of p*. The *region neighbor of p* minimizes $d_t(p,x)$, $x \in S \cap R$. We call a pair (p,q) a *region neighbor pair of S* if p is a region neighbor of q and vice versa.

We store a set L of some pairs of points. For a point $p \in S$ and a region R at p, the set L contains at most one pair (p,q), $q \in R$.

Definition. Let p be a point in $\mathbb{R}^k_t$ and R be a region at p and δ is a derived distance function. The region R is said to be *md-narrow* if, for any two points $x, y \in R$, $\delta(x) \leq \delta(y)$ implies $d_t(p,y) > d_t(x,y)$.

We use the construction of regions from the Section 3. The Lemma 4.1 gives $N_k = O(k)$ md-narrow regions.

4.1 Lemma. *Let $s = \langle s_1, \ldots, s_k \rangle$ be a simplex in $\mathbb{R}^k_t$ such that $\|s_i\| \geq 1$ and $\|s_i - s_j\| < 1/4$. Then the region corresponding to the simplex s is md-narrow.*

We shall describe the update algorithms. Let a heap H store the distances of the pairs of L. The heap item is the pair of the points. The key of the item (p, q) is the L_t-distance $d_t(p, q)$. The pair of points with minimal key is a closest pair of S.

With each point $p \in S$, we store a list $L(p) = \{q : (p, q) \in L\}$. The cardinality of $L(p)$ is at most N_k. With each point q in $L(p)$, we store a pointer to the item (p, q) of the heap H.

Denote the regions by $R(B_1, 0), \ldots, R(B_{N_k}, 0)$. We store N_k data structures $DS_1, \ldots,$
DS_{N_k}. The data structure DS_i corresponds to the region $R(B_i, 0)$. The data structure DS_i allows, for a query point p, to find a region neighbor q in i-th region $R(B_i, p)$ at p.

We need the following procedures. The procedure *insert_pair*(p, q)

1) insert the pair (p, q) into the heap H and

2) insert the points p and q into $L(q)$ and $L(p)$ respectively.

The procedure *delete_pair*(p, q)

1) delete the pair (p, q) from the heap H and

2) delete the points p and q from $L(q)$ and $L(p)$, respectively.

The insertion algorithm: Let p be a point to be inserted. Assume without loss of generality that S does not contain p. Create a list $L(p)$ (initially $L(p)$ is empty).

Insert p into DS_i for $i = 1, \ldots, N_k$. For $i = 1, \ldots, N_k$ perform the following:

1) Using the data structures DS_i, find the δ-region neighbor q of p in $S \cap R(B_i, p)$. If the δ-region neighbor of p doesn't exist then the processing of i-th region is completed and go to the next i.

2) Determine a region $R(B_j, q)$ at q which contains p.

3) If $L(q)$ does not contain a point in $R(B_j, q)$ then execute *insert_pair*(p, q) and go to the next i.

4) Let $L(q)$ contain the point $r \in R(B_j, q)$. If $d_t(p, q) \geq d_t(q, r)$ then go to the next i. Otherwise execute *delete_pair*(q, r).

5) Execute *insert_pair*(p, q).

The deletion algorithm: Let p be a point to be deleted. Delete p from DS_i for $i = 1, \ldots, N_k$. For each $q \in L(p)$ execute *delete_pair*(p, q). Delete the list $L(p)$. For $i = 1, \ldots, N_k$ perform the following steps.

1) Using the data structures DS_i, find the δ-region neighbor q of p in $S \cap R(B_i, p)$. If the δ-region neighbor of p doesn't exist then the processing of i-th region is completed and go to the next i.

2) For $j = 1, \ldots, N_k$ perform the following steps.

2.1) Using the data structures DS_j, find the δ-region neighbor r of q in $S \cap R(B_j, q)$. If the δ-region neighbor of q doesn't exist then the processing of j-th region is completed and go to the next j.

2.2) Determine a region $R(B_l, r)$ at r which contains q.

2.3) Let $L(r)$ contain the point $r' \in R(B_l, r)$. If $d_t(q, r) > d_t(r, r')$ then go to the next j. Otherwise execute *delete_pair*(r, r').

2.4) If $L(q)$ contains the point $q' \in R(B_j, q)$ then execute *delete_pair*(q, q').

2.5) Execute *insert_pair*(q, r).

We shall prove that the set $L = \{(p, q) : p \in S, q \in L(p)\}$ always contains a closest pair of S.

4.2 Lemma. *The set L contains the neighbor pairs of S after each update.*

4.3 Theorem. *Let some algorithm solve the dynamic problem of range searching for maximum in $U(k,n)$ update $Q(k,n)$ query and $P(k,n)$ preprocessing time, using $S(k,n)$ space. The dynamic problem of maintenance of a closest pair in $\mathbb{R}_t^k$ can be solved in $O(U(k,n)+Q(k,n)+k\log n)$ update and $c(k)(P(k,n)+O(kn))$ preprocessing time, using $c(k)(S(k,n))+O(kn))$ space. $c(k)$ is a number of md-narrow regions and depends on k only.*

Chazelle's data structure for range searching for maximum [2] allows to achieve query and update times of $O(\log^{k+1} n \log\log n)$, using $O(n\log^{k-2} n)$ space. Previously, no linear size data structure having polylogarithmic update time was known for maintenance of a closest pair of S in $\mathbb{R}^2$.

Unfortunately the update algorithms does not maintain the set of the neighbor pairs. We can increase the number of regions and modify the update algorithms to provide any maintened pair would be a region neighbor pair (and the set of maintened pairs would contain the neighbor pairs).

References

1. S. N. Bespamyatnykh, *Constructing Minimum Spanning Trees in $\mathbb{R}^k_\infty$ and Triangulation of the k-Cube*, manuscript.
2. B. Chazelle, *A Functional Approach to Data Structures and Its Use in Multidimensional Searching*, SIAM Journal on Computing **17** (1988), no. 3, 427–462.
3. H. N. Gabow, J. L. Bentley and R. E. Tarjan, *Scaling and Related Techniques for Geometry Problems*, Proc. 16-th Annual ACM Symposium on Theory of Computing (1984), 135–143.
4. J. G. Hocking and G. S. Young, *Topology*, Addison-Wesley, Reading, MA (1961).
5. S. Kapoor and M. Smid, *New Techniques for Exact and Approximate Dynamic Closest-Point Problems*, manuscript.
6. H.-P. Lenhof and M. Smid, *Enumerating the k Closest Pair Optimally*, Proc. 33rd Ann. IEEE Symp. Found. Comput. Sci. (1992), 380–386.
7. G. S. Lueker and D. E. Willard, *A Data Structure for Dynamic Range Queries*, Information Processing Letters (1982), no. 15, 209–213.
8. J. S. Salowe, *Enumerating Interdistances in Space*, Internat. J. Comput. Geom. Appl. (1992), no. 2, 49–59.
9. C. Schwarz, *Data Structures and Algorithms for the Dynamic Closest Pair Problem*, Ph.D. Thesis, Universität des Saarbrücken (1993).
10. M. Smid, *Maintaining the Minimal Distance of a Point Set in Less Than Linear Time*, Algorithms Rev. (1991), no. 2, 33–44.
11. M. Smid, *Maintaining the Minimal Distance of a Point Set in Polylogarithmic Time*, Discrete & Computational Geometry (1992), no. 7, 415–431.
12. K. L. Supowit, *New Techniques for Some Dynamic Closest-Point and Farthest-Point Problems*, Proc. 1-st Annual ACM-SIAM Symposium on Discrete Algorithms (1990), 84–90.
13. D. E. Willard, *Multidimensional Search Trees That Provide New Type of Memory Reductions*, Journal of the Association for Computing Machinery **34** (1987), no. 4, 846–858.
14. A. C. Yao, *On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems*, SIAM Journal on Computing (1982), no. 4, 721–736.

Ural State University
Department of Mathematics and Mechanics
51 Lenin St.
Ekaterinburg 620083
RUSSIA
E-mail: bsn@datakrat.urgu.e-burg.su