Data Structures for Range-Aggregate Extent Queries

Prosenjit Gupta* Ravi Janardan† Yokesh Kumar† Michiel Smid‡

Abstract

We consider a generalization of geometric range searching, with the goal of generating an informative "summary" of the objects contained in a query range via the application of a suitable *aggregation function* on these objects. We provide some of the first results for functions such as *closest pair*, *diameter*, and *width* that measure the extent (or "spread") of the retrieved set. We discuss a subset of our results, including closest pair queries on point-sets in the plane and on random point-sets in $\mathbb{R}^d$ ($d \geq 2$) and guaranteed-quality approximations for diameter and width queries in the plane, all for axes-parallel query rectangles.

1 Introduction

In a traditional instance of range searching, we are given a set, S, of geometric objects and wish to retrieve the subset S' contained in some query object Q (see [1] for a survey). Often, however, we desire a more informative "summary" of S', such as an order-statistic. (For instance, the average or the median price of homes (the "objects") in different neighborhoods (the "queries") of a city.) This can be done by applying, on S', an *aggregation function* such as *count*, *sum*, *min*, *max*, *mean*, *median*, *mode*, and *top-k* that is computed on a set of suitable weights defined on the objects (e.g., house prices). Prior work on such *range aggregate* query problems includes [3, 6, 8, 11, 15, 16, 17, 19, 18].

We present results for a new class of aggregation functions, including *closest pair*, *diameter*, and *width*, that measure the extent or "spread" of the objects in S'. Extent measures have applications in collision detection, shape-fitting, clustering etc. [2] and, instead of computing the measure on the entire set, it is often both sufficient and more efficient to "zoom in" on a query region and compute the measure only for this region (e.g., the closest pair of aircraft in a prescribed region of airspace).

A major challenge with extent functions is that (unlike, say, *count*) they are not decomposable efficiently, i.e., the answer for S' cannot be inferred quickly from answers for subsets that partition S'. (For instance, the closest pair in S' cannot be inferred in sublinear time from the closest pairs for subsets S'_1 and S'_2 that partition S.) Despite this, we obtain space- and query-time-efficient solutions (exact or guaranteed-quality approximations) to several range-aggregate extent queries, as summarized in Table 1. Our results are based on multilevel range trees, Voronoi Diagrams, Euclidean Minimum Spanning Trees, and generating sparse representations of candidate output sets and proving (expected) upper bounds on their size.

In prior related work, Shan *et al.* [12] gave empirical results for range-aggregate closest pair with axes-parallel query rectangles, based on R-trees. Gupta [7] gave a solution in $\mathbb{R}^1$ (resp., $\mathbb{R}^2$) with query time $O(1)$ using $O(n)$ space (resp., $O(\log^3 n)$ query time using $O(n^2 \log^3 n)$ space). Sharathkumar and Gupta [14] improved the 2D result to $O(\log^3 n)$ query time using $O(n \log^3 n)$ space and in [13, 14], showed how to decide in $O(\log^2 n)$ time and $O(n \log^{2+\epsilon} n)$ space if the closest pair in a query rectangle was within a user-specified tolerance. To our knowledge, there is no prior work on the range-aggregate diameter or width problems.

Due to space limitations, we present only results # 1, 2, 4, and 5 in Table 1 and omit proofs and most details.

2 Computing the closest pair in a query rectangle

We wish to preprocess a planar point-set S so that for a query rectangle Q, the closest pair in $S \cap Q$ can be reported. We develop our solution by successively generalizing solutions for simpler queries.

Computing the closest pair in a quadrant or vertical strip: First, let Q be a (north-east) quadrant. Let G be the graph with vertex set S where points p and q are connected by an edge iff (p, q) is the closest pair in $S \cap Q$ for some Q. G can be shown to be a plane graph and so has $O(n)$ edges, even though the number of "distinct" north-east quadrants (w.r.t. S) is $\Theta(n^2)$.

*Mentor Graphics, Hyderabad, and International Institute of Information Technology, Gachibowli, Hyderabad 500032, India. Supported, in part, by grants SR/S3/EECE/22/2004 and DST/INT/US/NSF-RPO-0155/04, Dept. of Science and Technology, Govt. of India. prosenjit_gupta@acm.org

†Dept. of Computer Science & Engg., Univ. of Minnesota, Minneapolis, MN 55455, U.S.A. {janardan,kumaryo}@cs.umn.edu. Supported, in part, by NSF grants INT–0422775 and CCF-0514950.

‡School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada. michiel@scs.carleton.ca. Research supported by NSERC.

#	Problem	Query	Space
1	Closest-pair in rect. (points in $\mathbb{R}^2$)	$\log^2 n$	$n \log^5 n$
2	Closest-pair in d-rect. (random points in $\mathbb{R}^d$)	$\log^{2d} n$	$n \log^{3d-2} n$ (expected)
3	Closest-pair in disk (random points in $\mathbb{R}^2$)	$n^{2/3+\epsilon}$ (expected)	$n^{1+\epsilon}$ (expected)
4	Diameter in rect. (points in $\mathbb{R}^2$)	$k \log^5 n$ $(1 \le k \le n)$	$(n + (n/k)^2) \log^2 n$
5	Approx. diameter in rect. (points in $\mathbb{R}^2$)	$\frac{1}{\sqrt{\delta}} \log^2 n$	$\frac{1}{\sqrt{\delta}} n \log n$
		$\frac{1}{\sqrt{\delta}} \log n + \log^3 n$ (variable δ)	$n \log^2 n$
6	Approx. width in rect. (points in $\mathbb{R}^2$)	$\frac{1}{\sqrt{\delta}} \log^3 n$	$\frac{1}{\sqrt{\delta}} n \log^2 n$
7	Closest-pair "partly in" hyper-rect. (points in $\mathbb{R}^d$)	$\log^d n$	$n \log^d n$
8	Closest-pair "partly in" halfspace (points in $\mathbb{R}^d$)	$n^{1-1/d+\epsilon}$	$n^{1+\epsilon}$
9	Closest-pair "partly in" ball (points in $\mathbb{R}^d$)	$n^{1-1/(d+1)+\epsilon}$	$n^{1+\epsilon}$

Table 1: Summary of results. Query rectangles are axes-parallel. "Random points" means that the points are chosen independently and uniformly at random in the unit-square. "Partly in" means that one point in the closest pair is in the query and the other is outside. Here k is a tunable parameter, $1 \le k \le n$, δ is an error tolerance parameter $0 < \delta < 1$, $\epsilon > 0$ is a constant, and $d \ge 2$ is a constant. "Variable δ" means that δ is part of the query; otherwise, it is fixed.

For each edge $e = (p, q)$ of G, we define the planar point $r_e = (\min(p_x, q_x), \min(p_y, q_y))$, with weight $d(p, q)$, where $d(\cdot, \cdot)$ is the Euclidean distance function. Let $R = \{r_e : e \text{ is an edge in } G\}$. Our problem is equivalent to reporting the point of minimum weight in $R \cap Q$, which can be done with a 2D range tree and fractional cascading.

Lemma 1 *A planar point-set S can be stored in a structure of size $O(n \log n)$ such that the closest pair in any north-east query quadrant Q can be reported in $O(\log n)$ time.*

For a query vertical strip, we will use the following:

Lemma 2 ([14]) *S can be stored in a structure of size $O(n \log^2 n)$ such that the closest pair in any vertical query strip can be reported in $O(\log n)$ time.*

The opposite-quadrant lemma: We can localize the closest-pair between points in two opposite quadrants as follows. Let A (resp. B) be the points of S strictly in the first (resp. third) quadrant Let $A_5 \subseteq A$ (resp. $B_5 \subseteq B$) be the $\min(5, |A|)$ (resp. $\min(5, |B|)$) points that are L_∞-closest to the origin.

Lemma 3 *Let (p, q) be the closest pair in S and let $p \in A$ and $q \in B$. Then, $p \in A_5$ and $q \in B_5$.*

Computing L_∞-neighbors in a quadrant: Assume S lies strictly in the first quadrant. Given the south-west quadrant Q_q of a query point q in the first quadrant, we wish to report the $\min(5, |S \cap Q_q|)$ points in $S \cap Q_q$ that are L_∞-closest to the origin.

Let A be the set of points of S on or below the diagonal $y = x$, and let $B := S \setminus A$. Then, for each point p in A (resp. B), the L_∞-distance between p and the origin is equal to the x-coordinate (resp. y-coordinate) of p. This leads to the following structure to answer queries when q is, wlog, on or above the diagonal $y = x$: We maintain (i) an array storing the points of A sorted by x-coordinates, and (ii) an array storing the points of B sorted by x-coordinates; with each entry p in this array, we store the $\min(5, |B_p|)$ lowest points in B_p, where $B_p := \{b \in B : b_x \le p_x\}$.

Lemma 4 *A set S of points strictly in the first quadrant can be stored in a structure of size $O(n)$ such that for any query point q strictly in the first quadrant, the $\min(5, |S \cap Q_q|)$ points in $S \cap Q_q$ that are L_∞-closest to the origin can be reported in $O(\log n)$ time.*

Computing the closest pair in an anchored 3-sided rectangle: Let ℓ be a fixed vertical line. An *anchored 3-sided rectangle* Q is a rectangle of the form $Q = [a, b] \times [c, \infty)$ that intersects ℓ. Given Q, we wish to report the closest pair in $S \cap Q$.

Let T be a balanced binary search tree storing S at its leaves, by y-order. Let u be the highest node on the right spine of T such that the horizontal line ℓ'_u that separates the left and right subtrees of u intersects Q. Point $X_u = \ell \cap \ell'_u$ partitions the plane into four quadrants. Let u_1 and u_2 be the left and right children of u, respectively. Let $S^l_{u_1}$ (resp. $S^r_{u_2}$) be the points of S_{u_1} to the left (resp. right) of ℓ. (Throughout, S_v denotes the subset of S stored at the leaves of v's subtree.) Define $S^l_{u_2}$ and $S^r_{u_2}$ similarly w.r.t S_{u_2}.

Six cases exist for the closest pair (p, q) in $S \cap Q$. (1) p and q are both to the left of ℓ: Then (p, q) is the closest pair of $S^l_{u_1} \cup S^l_{u_2}$ which is in the north-east quadrant of the point (a, c). We find (p, q) by storing at u the structure of Lemma 1 for $S^l_{u_1} \cup S^l_{u_2}$. (2) p and q are both to the right of ℓ: This is is symmetric to (1). (3) p and q are both above ℓ'_u: Then (p, q) is the closest pair of $S^l_{u_2} \cup S^r_{u_2}$ which is in the strip bounded by the vertical lines through (a, c) and (b, c). We find (p, q) by storing at u the structure of Lemma 2 for $S^l_{u_2} \cup S^r_{u_2}$. (4) p (resp. q) is in the south-west (resp. north-east) quadrant of X_u: By storing at u appropriate variants of the structure of Lemma 4, and using Lemma 3, we can compute 25 point-pairs, such that (p, q) is among them. (5) p (resp. q) is in the north-west (resp. south-east) quadrant of X_u: This is symmetric to (4). (6) p and q are both below ℓ'_u: Then both points are in the subtree of u_1. We can find (p, q) by recursively querying this subtree.

Lemma 5 *S can be stored in a structure of size $O(n \log^3 n)$ the closest pair in any anchored 3-sided query rectangle Q can be reported in $O(\log^2 n)$ time.*

Computing the closest pair in an anchored rectangle: Let ℓ be a fixed horizontal line. An *anchored rectangle* Q is a rectangle of the form $[a, b] \times [c, d]$ that intersects ℓ. Given Q, we wish to report the closest pair in $S \cap Q$.

Let T be a balanced binary search tree storing S at its leaves, by x-order. Using T, we can reduce our query to four queries for anchored 3-sided rectangles and two closest pair queries for opposite quadrants.

Lemma 6 *S can be stored in a structure of size $O(n \log^4 n)$ such that the closest pair in any anchored query rectangle Q can be reported in $O(\log^2 n)$ time.*

General closest pair rectangle queries: Given a general query rectangle Q, we wish to report the closest pair in $S \cap Q$.

Let T be a balanced binary search tree storing S at its leaves, by y-order. For each internal node u of T, define the horizontal line ℓ'_u as before. We store at u the structure of Lemma 6 for S_u, to answer closest pair queries for rectangles anchored w.r.t. ℓ'_u.

Given Q, we search down T to the first node u such that ℓ'_u intersects Q. Q is anchored w.r.t. ℓ'_u, so we use the structure for S_u to find the closest pair in $S \cap Q$.

Theorem 7 *A set S of n points in the plane can be stored in a structure of size $O(n \log^5 n)$ such that for any axes-parallel query rectangle Q, the closest pair in $S \cap Q$ can be reported in $O(\log^2 n)$ time.*

3 Closest pair rectangle queries on randomly distributed points

Let S be a set of n points in the plane, chosen independently and uniformly at random in the unit-square. We obtain a data structure of expected size $O(n \log^4 n)$ and query time $O(\log^4 n)$ time. Though not as efficient, asymptotically, as the one in [14], our solution is simple and practical, and, moreover, extends naturally to any fixed dimension $d > 2$.

Our approach is to precompute each point-pair (p, q), with $p, q \in S$, that is the closest pair for at least one query rectangle. We then store each such pair as a weighted point in a four-dimensional range tree. The four dimensions are the x- and y-coordinates of p and of q; the weight is the Euclidean distance $d(p, q)$. Given a query rectangle Q, we find the closest pair in $S \cap Q$ by doing a range-minimum query [5] on the tree with the hyper-rectangle $Q \times Q$.

Formally, let $\mathcal{Q}$ be the (infinite) set of all axes-parallel query rectangles. Let Λ be the number of pairs (p, q), with $p, q \in S$ and p to the left of q, such that there is a rectangle $Q \in \mathcal{Q}$ for which (p, q) is a closest pair in $S \cap Q$. Then our structure uses $O(\Lambda \log^3 n)$ space and has a query time of $O(\log^4 n)$. Moreover, it can be built in time equal to that needed to compute the Λ pairs plus $O(\Lambda \log^3 n)$ time. Thus, if Λ is "small", then this will be an efficient and practical solution.

Unfortunately, Λ can be $\Theta(n^2)$ in the worst case. (Take two sets of $n/2$ points on the boundary of the unit-circle, in opposite quadrants. Every pair of points, one from each set, contributes 1 to Λ.) However, if the points of S are chosen at random then the expected value of Λ is $O(n \log n)$, as seen below.

Lemma 8 *Let (p, q) be an ordered point-pair, with $p, q \in S$ and p to the left of q. This pair contributes 1 to Λ iff the rectangle, $R(p, q)$, that has $\overline{pq}$ as a diagonal is empty, i.e., contains no point of $S \setminus \{p, q\}$.*

Thus, Λ is the number of empty rectangles $R(p, q)$ in Lemma 8. If the points of S are chosen at random, then they are "well-distributed" and there will not be many empty rectangles $R(p, q)$, as formalized by Lemma 9. (This result also appears, without proof, in [4].)

Lemma 9 *For a set S of n points that are chosen independently and uniformly at random in the unit-square, the expected value, $E(\Lambda)$, of Λ is $O(n \log n)$.*

Thus, the closest pair in $S \cap Q$ can be computed in $O(\log^4 n)$ time using a structure of expected size $O(n \log^4 n)$.

This approach generalizes to $\mathbb{R}^d$, for any fixed $d \geq 3$, based on a result from [10] that there are $O(n \log^{d-1} n/(d-1)!)$ so-called direct domination pairs (p, q) among n points drawn independently at random from the unit-hypercube in $\mathbb{R}^d$, since each such pair defines the diagonal of an empty hyper-rectangle. Thus Λ is $O(n \log^{d-1} n)$. Our problem reduces to storing each of the Λ pairs (p, q) as a weighted point in a $2d$-dimensional range tree. We conclude:

Theorem 10 *A set S of n points chosen independently and uniformly at random in the unit-hypercube in $\mathbb{R}^d$, $d \geq 2$, can be stored in a structure of expected size $O(n \log^{3d-2} n)$ so that the closest pair in any axes-parallel query rectangle can be reported in $O(\log^{2d} n)$ time.*

4 Approximating the diameter in a query rectangle

Let S be a set of n points in the plane and let δ be a fixed real, $0 < \delta < 1$. Given a query rectangle Q, we wish to report a pair of points in $S \cap Q$ whose distance is at least $(1 - \delta)$ times the diameter of $S \cap Q$.

Let $\beta(\delta) = \left\lceil \frac{2 \arcsin(1-\delta)}{\pi - 2 \arcsin(1-\delta)} \right\rceil$; $\beta(\delta) = O(1/\sqrt{\delta})$ if δ converges to zero. Our approach uses the following:

Lemma 11 ([9]) *Choose $2(\beta(\delta)+1)$ equally-spaced vectors around the unit-circle. For each vector d_i, let p_i (resp. q_i) be the point of S that is extreme in direction d_i (resp. $-d_i$). Let D be the diameter of S and let $\Delta = \max_i d(p_i, q_i)$. Then $1-\delta \leq \Delta/D \leq 1$.*

Our structure is a 2D range tree on S. With each node v in each secondary tree, we store the $O(\beta(\delta)) = O(1/\sqrt{\delta})$ point-pairs of Lemma 11 for S_v. The space used is $O((1/\sqrt{\delta})n\log n)$.

Given Q, we compute a set C of $O(\log^2 n)$ canonical nodes v in the secondary structures of the range tree such that $S \cap Q = \cup_{v\in C} S_v$. Consider any of the $O(\beta(\delta))$ direction pairs d_i and $-d_i$. We compute the extreme points of $S\cap Q$ in directions d_i and $-d_i$ by computing the extreme points among those stored with the canonical nodes for this direction pair. By Lemma 11, the farthest pair so computed over all direction pairs is an approximation to the diameter of $S \cap Q$.

Theorem 12 *A set S of n points in the plane can be stored in a structure of size $O((1/\sqrt{\delta})n\log n)$, so that for any axes-parallel query rectangle Q, a $(1-\delta)$-approximation to the diameter of $S\cap Q$ can be reported in $O((1/\sqrt{\delta})\log^2 n)$ time, where $0<\delta<1$.*

This solution extends to queries where δ comes as an input parameter along with Q. The idea is to use the range tree on S to also compute the convex hull of the $S\cap Q$ (by repeated merging of convex hulls stored at the canonical nodes) and then finding the extremal points for each of the $O((1/\sqrt{\delta})$ directions—in logarithmic time per merge and per direction. This yields the bounds shown in Table 1.

5 Approximating the width in a query rectangle

The *width* of a planar point-set S is the width of a narrowest enclosing strip. For a query rectangle Q, we wish to report a strip enclosing $S\cap Q$ of width at most $(1+\delta)$ times the width of $S\cap Q$, for a fixed real δ, $0<\delta<1$.

Let $\gamma(\delta) = \left\lceil \frac{\pi}{2\arccos(1/(1+\delta))} \right\rceil$; $\gamma(\delta) = O(1/\sqrt{\delta})$ if δ converges to zero. Our approach uses the following:

Lemma 13 ([9]) *Let $S_0 = S$ and S_i be a copy of S rotated clockwise around the origin from S_{i-1} by an angle $\pi/\gamma(\delta)$, $1 \leq i \leq \gamma(\delta)$. For $0 \leq i \leq \gamma(\delta)$, let L_i (resp. R_i) be the downward (resp. upward) convex chain dual to the upper (resp. lower) hull of the convex hull of S_i. Let ω_i^L (resp. ω_i^R) be the minimum distance between any vertex of L_i (resp. R_i) and any point vertically below (resp. above) it on R_i (resp. L_i). Let $\Omega = \min_i\{\omega_i^L, \omega_i^R\}$ and let W be the width of S. Then $1 \leq \Omega/W \leq 1+\delta$.*

Both ω_i^L and ω_i^R can be computed in $O(\log^2 n)$ time if L_i and R_i are stored in balanced binary search trees [9].

We store S in a range tree. Each node v in each secondary tree stores $1+\gamma(\delta)$ instances of the structure of Lemma 13 for S_v, where the i-th instance is a pair of balanced binary search trees built on the dual chains $L_i(v)$ and $R_i(v)$ associated with the i-th rotated copy of S_v. The space is $O((1/\sqrt{\delta})n\log^2 n)$.

Given Q, we compute, in $O(\log^2 n)$ time, a set C of $O(\log^2 n)$ canonical nodes v in the secondary structures such that $S\cap Q = \cup_{v\in C} S_v$. For each i, we merge the $L_i(v)$'s for all $v\in C$ into a single chain in $O((1/\sqrt{\delta})\log^3 n)$ time. Similarly, for the $R_i(v)$'s. From these $O(1/\sqrt{\delta})$ pairs of chains, we compute the minimum vertical distance between each pair and take the smallest as Ω.

Theorem 14 *A set S of n points in the plane can be stored in a structure of size $O((1/\sqrt{\delta})n\log^2 n)$ such that for any query rectangle Q, a $(1+\delta)$-approximation to the width of $S\cap Q$ can be reported in $O((1/\sqrt{\delta})\log^3 n)$ time, where $0<\delta<1$.*

References

[1] P. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 1–56. AMS, 1999.

[2] P. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51:606–635, 2004.

[3] P. Bose, E. Kranakis, P. Morin, and Y. Tang. Approximate range mode and range median queries. In *Proc. 22nd Symp. on Theoretical Aspects of Computer Science*, volume 3404 of *LNCS*, pages 377–388, Berlin, 2005. Springer-Verlag.

[4] S. Felsner. Empty rectangles and graph dimension. arXiv:math/0601767v1, 2006. http://arxiv.org/abs/math/0601767v1.

[5] H. Gabow, J.Bentley, and R. Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th ACM Symp. on Theory of Computing*, pages 135–142, 1984.

[6] S. Govindarajan, P. Agarwal, and L. Arge. CRB-tree: An efficient indexing scheme for range aggregate queries. In *Proc. 9th Intl. Conf. on Database Theory*, pages 143–157, 2003.

[7] P. Gupta. Algorithms for range-aggregate query problems involving geometric aggregation operations. In *Proc. 16th Intl. Symp. on Algorithms and Computation*, volume 3827 of *LNCS*, pages 892–901, Berlin, 2005. Springer-Verlag.

[8] S. Hong, B. Song, and S. Lee. Efficient execution of range-aggregate queries in data warehouse environments. In *Proc. 20th Intl. Conf. on Conceptual Modeling*, volume 2224 of *LNCS*, pages 299–310, Berlin, 2001. Springer-Verlag.

[9] R. Janardan. On maintaining the width and diameter of a planar point-set online. *Intl. Journal of Computational Geometry & Applications*, 3:331–344, 1993.

[10] R. Klein. Direct dominance of points. *Intl. Journal of Computer Mathematics*, 19:225–244, 1987.

[11] D. Krizanc, P. Morin, and M. Smid. Range mode and range median queries on lists and trees. *Nordic Journal of Computing*, 12:1–17, 2005.

[12] J. Shan, D. Zhang, and B. Salzberg. On spatial-range closest-pair query. In *Proc. 8th Intl. Symp. on Spatial and Temporal Databases*, volume 2750 of *LNCS*, pages 252–269, Berlin, 2003. Springer-Verlag.

[13] R. Sharathkumar and P. Gupta. Range-aggregate proximity detection for design rule checking in VLSI layouts. In *Proc. 18th Canadian Conf. on Computational Geometry*, pages 151–154, 2006.

[14] R. Sharathkumar and P. Gupta. Range-aggregate proximity queries. IIIT/TR/2007/80, Intl. Inst. of Info. Tech., Hyderabad, www.iiit.net/techreports/2007_80.pdf, 2007.

[15] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2002.

[16] Y. Tao and D. Papadias. Range aggregate processing in spatial databases. *IEEE Trans. on Knowledge and Data Engineering*, 16:1555–1570, 2004.

[17] D. Zhang, A. Markowetz, V. Tsotras, D. Gunopulos, and B. Seeger. Efficient computation of temporal aggregates with range predicates. In *Proc. 20th Symp. on Principles of Database Systems*, pages 237–245, 2001.

[18] D. Zhang, V. Tsotras, and D. Gunopulos. Efficient aggregation over objects with extent. In *Proc. 21st Symp. on Principles of Database Systems*, pages 121–132, 2002.

[19] D. Zhang and V. J. Tsotras. Improving min/max aggregation over spatial objects. *VLDB Journal*, 14:170–181, 2005.