Closest-Point Problems Simplified on the RAM

Timothy M. Chan*

Basic proximity problems for low-dimensional point sets, such as closest pair (CP) and approximate nearest neighbor (ANN), have been studied extensively in the computational geometry literature, with well over a hundred papers published (we merely cite the survey by Smid [10] and omit most references). Generally, optimal algorithms designed for worst-case input require hierarchical spatial structures with sophisticated balancing conditions (we mention, for example, the BBD trees of Arya *et al.*, balanced quadtrees, and Callahan and Kosaraju's fair-split trees); dynamization of these structures is even more involved (relying on Sleator and Tarjan's dynamic trees or Frederickson's topology trees).

In this note, we point out that much simpler algorithms with the same performance are possible using standard, though nonalgebraic, RAM operations. This is interesting, considering that nonalgebraic operations have been used before in the literature (e.g., in the original version of the BBD tree [2], as well as in various randomized CP methods).

The CP algorithm can be stated completely in one paragraph. Assume coordinates are positive integers bounded by $U = 2^w$. Given a point p in a constant dimension d where the i-th coordinate p_i is the number $p_{iw} \cdots p_{i0}$ in binary,

- define its *shuffle* $\sigma(p)$ to be the number $p_{1w} \cdots p_{dw} \cdots p_{10} \cdots p_{d0}$ in binary, and
- define *shifts* $\tau_i(p) = (p_1 + \lfloor i2^w/(d+1) \rfloor, \ldots, p_d + \lfloor i2^w/(d+1) \rfloor)$ for $i = 0, \ldots, d$, assuming w.lo.g. that d is even.

For each $i = 0, \ldots, d$, sort the n given points p according to $\sigma(\tau_i(p))$. Then a constant-factor approximate CP can be found among the adjacent pairs in these $d+1$ lists. Furthermore, the exact CP can be found among all pairs at most c positions apart in these lists, where c is a sufficiently large constant (depending exponentially on d). Correctness can be proved from two observations: points inside a "quadtree box" appear consecutively in the shuffle order [3], and every pair of points of distance r lies in a quadtree box of diameter $O(r)$ after some shift [6].

This algorithm is not original. The approximate version was most recently proposed by Lopez and Liao [7, 8], although sorting along shuffle order, or generally space-filling curves, has been suggested often in other applied areas such as databases and pattern recognition. In computational geometry, it was used by Bern *et al.* [3] (for constructing unbalanced and balanced quadtrees, which we are trying to do without here), but is largely overlooked as a theoretical tool (hence the reason for writing this note). Shifting, on the other hand, is a well-known technique in approximation.

One objection is that we can't directly compute $\sigma(p)$. But we can decide whether $\sigma(p) < \sigma(q)$ by a straightforward procedure:

$$\begin{array}{l} i \leftarrow 1; \\ \text{for } j = 2, \ldots, d \text{ do} \\ \qquad \text{if } |p_i \oplus q_i| < |p_j \oplus q_j| \text{ then } i \leftarrow j; \\ \text{return } p_i < q_i. \end{array}$$

Here $\oplus$ denotes bitwise exclusive-or and $|x|$ denotes $\lfloor \log_2 x \rfloor$. Though not realized in previous papers [3, 8], we actually don't need extra primitives to compute $|\cdot|$, as we can decide whether $|x| < |y|$ by this neat trick:

$$\text{if } x > y \text{ then return false else return } x < x \oplus y.$$

So, using comparison-based sorting, the entire algorithm can be implemented in $O(n \log n)$ time with only the bitwise exclusive-or!

Although there are randomized CP algorithms matching in simplicity, this CP algorithm is remarkable in that it can be dynamized *automatically*, since each sorted list—and hence its pairs of positions $\leq c$ apart—can be updated in logarithmic time by standard search trees (we can use a heap to store these pairs). Contrast this with the previous, far more complicated, $O(\log n)$ dynamic CP method of Bespamyatnikh [4] or its (many!) predecessors.

The same $d+1$ lists give solutions to ANN queries with a constant approximation factor [7, 8], which can be refined to $1+\varepsilon$ by "scattering" $O(1/\varepsilon^d)$ query points around the given point. So, ANN queries can be answered in $O((1/\varepsilon^d) \log n)$ time with $O(\log n)$ update time. In fact, a fancier modification yields query time

*Dept. of Computer Science, Univ. of Waterloo, Waterloo, Ontario N2L 3G1, Canada, tmchan@uwaterloo.ca. Work supported in part by an NSERC Research Grant.

$O(\log n + 1/\varepsilon^d)$, improving the original result by Arya *et al.* [2]: Define $\ell(p,q) = \max_j |p_j \oplus q_j|$. In list i, assign every point p, with predecessor p^-, the *level* number $\ell(\tau_i(p^-), \tau_i(p))$. Given query point q, search for its successor p. Check the next c_ε points after q of levels exceeding $\ell(\tau_i(p), \tau_i(q)) - b_\varepsilon$, for suitable constants $c_\varepsilon = \Theta(1/\varepsilon^d)$ and $b_\varepsilon = \Theta(\log(1/\varepsilon))$. These points can be reported in $O(\log n + c_\varepsilon)$ time by a priority search tree. Similarly, check c_ε points in the reverse list. After all lists have been examined, a $(1+\varepsilon)$-factor ANN will be found. Again, only the exclusive-or operation is needed.

The same ideas can be applied to other proximity problems like minimum spanning trees and $(1+\varepsilon)$-spanners. Because the approach directly reduces d-dimensional proximity problems to 1-dimensional sorting and searching, it effortlessly yields efficient parallel or external-memory algorithms.

We can also surpass algebraic lower bounds by exploiting the full power of the word RAM. Rabin's randomized linear-time algorithm for CP was perhaps the earliest such example. For the case $U = n^{O(1)}$ (i.e., points on a polynomial-size grid), we have immediately a deterministic linear-time algorithm for CP, if we use radix-sort instead of a comparison sort (we can precompute shuffles of points of logarithmic length in linear time and store them in a table); this result is new and improves a previous $O(n \log\log n)$ deterministic algorithm of Reif and Tate [9]. For any U, assuming the shuffle operation (trivially AC^0) is built in, we can solve the CP problem by directly applying the current deterministic integer-sorting results on the word RAM. We can also solve the dynamic CP problem or answer ANN queries in $O(\min\{\log\log U, \sqrt{\log n}\})$ time on the RAM, by directly using van Emde Boas trees or Andersson's exponential search trees; this simplifies and extends a recent work of Amir *et al.* [1], who had to explicitly generalize van Emde Boas trees to get $O(\log\log U)$ algorithms for ANN and incremental CP only. Similarly, the static structure in another recent paper of Cary [5] is simplified using the shuffle operation.

References

[1] A. Amir, A. Efrat, P. Indyk, and H. Samet. Efficient regular data structures and algorithms for location and proximity problems. In *Proc. 40th IEEE Sympos. Found. Comput. Sci.*, pages 160–170, 1999.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. ACM*, 45:891–923, 1998.

[3] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *Int. J. Comput. Geom. Appl.*, 9:517–532, 1999.

[4] S. N. Bespamyatnikh. An optimal algorithm for closest pair maintenance. *Discrete Comput. Geom.*, 19:175–195, 1998.

[5] M. Cary. Towards optimal ϵ-approximate nearest neighbor algorithms in constant dimensions. *J. Algorithms*, to appear. `http://www.cs.washington.edu/homes/cary/papers/nn.ps.gz`.

[6] T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comp. Geom.*, 20:359–373, 1998.

[7] S. Liao, M. A. Lopez, and S. T. Leutenegger. High dimensional similarity search with space filling curves. In *Proc. of Int. Conf. on Data Engineering*, 2001. `http://www.cs.du.edu/~leut/icde01.ps`.

[8] M. A. Lopez and S. Liao. Finding k-closest-pairs efficiently for high dimensional data. In *Proc. 12th Canad. Conf. Comput. Geom.*, pages 197–204, 2000. `http://www.cs.unb.ca/conf/cccg/eProceedings/29.ps.gz`.

[9] J. H. Reif and S. R. Tate. Fast spatial decomposition and closest pair computation for limited precision input. *Algorithmica*, 28:271–287, 2000.

[10] M. Smid. Closest-point problems in computational geometry. In *Handbook of Computational Geometry* (J. Urrutia and J. Sack, ed.), North-Holland, pages 877–935, 2000.