Normalized divide and conquer: A scaling technique for solving multi-dimensional problems

Rolf G. Karlsson and Mark H. Overmars

RUU-CS-86-19
oktober 1986

Normalized divide and conquer: A scaling technique for solving multi-dimensional problems

Rolf G. Karlsson and Mark H. Overmars

RUU-CS-86-19
oktober 1986

Department of Computer Science
University of Utrecht
P.O. Box 80.012
3508 TA Utrecht
the Netherlands

Normalized divide and conquer: A scaling technique for solving multi-dimensional problems

Rolf G. Karlsson and Mark H. Overmars

Abstract. A new technique, named *normalized divide and conquer* is presented to solve multi-dimensional problems on a grid in a very efficient way. It is also shown that a number of problems concerning objects in arbitrary space can be transformed into problems on a grid by using another form of normalization. In this way new solutions are obtained for the multi-dimensional maximal elements and rectangle intersection problem that are more efficient than previous solutions.

Keywords and phrases. Normalized divide and conquer, scaling techniques, multi-dimensional problems, maximal elements, rectangle problems.

1. Introduction.

Computational geometry studies the computational complexity of finite geometric problems. Recently there has been a growing interest in solving problems in computational geometry on a grid, i.e. points (or end points of line segments, vertices of polygons, etc.) are in $U^d = [0..u-1]^d$ rather than having arbitrary real coordinates. See for example Karlsson[6], Karlsson and Munro[7], Karlsson and Overmars[8,9], Keil and Kirkpatrick[10], Müller[11], Overmars[12,13] and Willard[14,15].

On a grid many problems can be solved more efficiency by using perfect hashing techniques or table look-up but, because they require large amounts of preprocessing, such methods will not work for the problems we consider. Instead, we only use some restricted forms of table look-up to achieve our efficient results.

Up to now mainly problems on 1- and 2-dimensional grids have been considered. We will develop a new technique for solving multi-dimensional problems on a grid, named *normalized divide and conquer*. Briefly, normalized divide and conquer is a divide and conquer technique that splits the multi-dimensional universe in equal halves, rather than splitting the set of objects in equal halves. To obtain the desired reduction in size of the universe after a divide step we use a normalization technique.

Authors' addresses:
Rolf G. Karlsson: Dept. of Computer Science, Linköping University, 581 83 Linköping, Sweden.
Mark H. Overmars: Dept of Computer Science, University of Utrecht, P.O.Box 80.012, 3508 TA Utrecht, the Netherlands.

Gabow, Bentley and Tarjan[3], recently proposed similar ideas, and for one of the problems we consider, the maximal elements problem, they provided the same upper-bound. For the other problems considered in this paper our techniques provide faster solutions. Our solutions also display a desirable simplicity.

We will demonstrate the new technique by applying it to the following problems:

Maximal elements: Given a set of points on a d-dimensional grid, determine those points that are maximal.

Rectangle intersections: Given a set of axis-parallel hyper-rectangles on a d-dimensional grid, determine all pairs of hyper-rectangles that intersect. Two types of intersections are considered and the efficiency of the solution depends on the type chosen. *Boundary intersection:* Two rectangles intersect if their boundaries intersect. *True intersection:* Two rectangles intersect if they have a point in their interior in common, i.e., either their boundaries intersect or one is completely contained in the other.

By first normalizing problems in arbitrary space to problems on a grid the bounds we obtain also apply to the problems in arbitrary space. The following table displays the results obtained (all logarithms to the base 2).

Problem	Known bound	New bound
maximal elements	$n\log^{d-3}n$ loglog n [3]	$n\log^{d-3}n$ loglog n
boundary intersections	$k + n\log^{d-1}n$ [2,3]	$k + n\log^{d-2}n$ loglog n
true intersections	$k + n\log^{d-1}n$ [2,3]	$k + n\log^{d-1}n/\text{loglog } n$

The paper is organized as follows. In Section 2 we recall some known results that will be used in the rest of the paper. In Section 3 we demonstrate the technique of normalized divide and conquer by applying it to solve the d-dimensional maximal elements problem efficiently. In Section 4 we concentrate on the rectangle intersection problem. We first generalize a divide and conquer technique by Güting[4,5] for solving the rectangle intersection problem in 2-dimensional space to multi-dimensional space. Next we use normalized divide and conquer to improve the time bounds. Finally, in Section 5, we give some concluding remarks and mention some open problems.

2. Preliminaries.

Let us first recall a number of known results concerning problems on 2- and 3-dimensional grids that will be used throughout this paper. The next three theorems state results for the low-dimensional versions of the problems we will consider in multi-dimensional space. Our multi-dimensional problems will eventually be reduced to these low-dimensional versions. All three results can be found in Karlsson and Overmars[8].

Theorem 2.1. *Given a set V of n points from U^3, the set of maximal elements among V can be computed in time $O(n\,\text{loglog}\,u)$ using $O(n+u)$ storage.*

The result is obtained by using a space sweep technique. We sweep a plane over the 3-dimensional space, keeping track of the contour of maximal elements found. A new point encountered is tested with respect to this contour and, if maximal, added to the contour. Because the points lie on a grid such a test can be performed in time O(loglog u). For details see [8]. As in this paper applications u will be O(n) the theorem states that maximal elements in 3-dimensional space can be found in O(n loglog n) time using O(n) storage.

Theorem 2.2. *Given a set V of n axis-parallel rectangles on U^2, all pairs of rectangles in V that boundary intersect can be located in time O(k + n loglog u) using O(n+u) storage, where k is the number of reported answers.*

This result is obtained by reducing the boundary intersection problem to the problem of finding all intersections in a set of horizontal and vertical line segments. A scanline technique is used in which we scan the grid with a vertical line from left to right, maintaining the intersections with horizontal line segments. When a vertical line segment is encountered a range query is performed. Due to the fact that points lie on a grid, such a range query takes time O(loglog u). (For details see [8].) Again in our applications we can replace u by n.

Theorem 2.3. *Given a set V of n axis-parallel rectangles on U^2, all pairs of rectangles in V that true intersect can be located in time O(k + n log u/loglog u) using O(n+u) storage, where k is the number of reported answers.*

The true intersection problem is split into boundary intersections and inclusions. The boundary intersection problem is solved as stated above. The inclusion problem is again solved using a scanline algorithm, exploiting a new data structure called an interval trie (see [8]).

3. Maximal elements.

In this section we will demonstrate the technique of normalized divide and conquer by applying it to the d-dimensional maximal elements problem on a grid. To solve the problem we will treat a more general problem: Given two sets of points V_1 and V_2, compute all points in V_1 that are maximal with respect to V_2. (A point $p=(p_1,p_2,\cdots,p_d)$ is said to be maximal with respect to a set V if there is no point $q=(q_1,q_2,\cdots,q_d)\in V$ such that $p_i \le q_i$ for all i.) A solution to this more general problem can easily be used for solving the normal problem by taking $V_1=V_2=V$.

The 3-dimensional version of the generalized problem can easily be solved in time O(n loglog u) using the technique in [8]. With this result as a basis we will consider higher dimensions.

Let V_1 and V_2 be two sets of points in an arbitrary d-dimensional space. For convenience, we assume that no two points have the same value in some coordinate. Such degeneracies can easily be solved by bucketing points with equal value in some coordinate (only one of them is relevant in finding maximal elements). We omit tedious details. To find the maximal elements in V_1 with respect to V_2 we first reduce the

problem to a grid. To do so we *normalize* the points in V_1 and V_2 by replacing the point $p=(p_1, \cdots, p_d)$ by $R(p)=(R_1(p_1), \cdots, R_d(p_d))$ where $R_i(p_i)$ is the rank of p_i with respect to the i-th coordinates of the points in V_1 and V_2. It is easy to see that this normalization preserves the maximality property, i.e., p is maximal with respect to V_2 if and only if $R(p)$ is maximal with respect to $R(V_2)$, the set of all normalized points in V_2.

To solve the maximal elements problem on a d-dimensional grid we use normalized divide and conquer. We split the grid in two equal halves with respect to the d-th coordinate. Let A_1 and A_2 be the subsets of V_1 and V_2 that lie in the left half (i.e., with small d-th coordinate) and let B_1 and B_2 be the subsets of V_1 and V_2 that lie in the right half. Clearly, the points in B_1 are maximal with respect to V_2 if and only if they are maximal with respect to B_2. Hence, we can determine the maximal elements by recursively solving the problem on the right half (see below). Points in A_1 might not be maximal because they are dominated by points in B_2. So, before going into recursion, we have to remove those points in A_1 that are dominated by a point in B_2. To this end we project both A_1 and B_2 on the separating hyperplane and solve the maximal elements problem on the resulting $d-1$ dimensional grid, eliminating from A_1 all points that are not maximal with respect to B_2.

The next step is crucial. We normalize the sets A_1, A_2, B_1 and B_2 with respect to the half they are in, i.e., any point $p=(p_1, \cdots, p_d)$ in A_1 or A_2 we replace by $R(p)=(R_1(p_1), \cdots, R_d(p_d))$ where $R_i(p_i)$ is the rank of p_i with respect to the i-th coordinate of the points in A_1 and A_2 (similar for B_1 and B_2). Because of our assumption that no two points lie on the same grid line, this reduces the universe in both halves to $(n/2)^d$.

After the splitting, reduction of A_1 and normalization, we recursively compute the maximal elements of A_1 with respect to A_2 and of B_1 with respect to B_2.

Note that this technique is similar to known techniques for solving the multi-dimensional maximal elements problems (see e.g. Bentley[1]), except that we split the universe rather than the point set and use the normalization step. As a result, once we get to low-dimensional problems, we know that our points lie on a grid and, hence, can use fast solutions.

Theorem 3.1. *The d-dimensional maximal elements problem ($d>3$) can be solved in time $O(n \log^{d-3} n \ \text{loglog}\ n)$.*

Proof. The first normalization to transform the general problem to a grid problem takes time $O(n \log n)$. Let $T(d,n)$ denote the amount of time required for solving the d-dimensional maximal elements problem on a grid of size n^d. We know

$$T(3,n) = O(n \,\text{loglog}\, n)$$

Moreover, from the algorithm above it follows that

$$T(d,n) = 2T(d,n/2) + T(d-1,n) + O(n)$$

This immediately leads to

$$T(d,n) = O(n \log^{d-3} n \ \text{loglog}\ n)$$

□

Note that the normalization step is crucial. If we had omitted this step, a divide would only reduce the universe to $n^d(n/2)$. This would not yield efficient time bounds.

4. Rectangle intersection.

We will now apply normalized divide and conquer to solve the multi-dimensional rectangle intersection problem efficiently. First we concentrate on the true intersection problem. Later we show how to improve the method when only boundary intersections are required. Known solutions for the multi-dimensional rectangle intersection problem reduce it to a query problem. To be able to use normalized divide and conquer we have to design a divide and conquer approach. To this end we generalize a method by Güting[4] (see also [5]). In [4] the 2-dimensional rectangle intersection problem is solved using divide and conquer. We will describe a multi-dimensional version of this method.

Again we generalize the problem. Rather that having one set of rectangles, we assume we have two sets of rectangles, V_1 and V_2, and we ask for all intersections between elements in V_1 and V_2. Clearly, we can use a solution to this more general problem for solving the original problem by choosing $V_1=V_2=V$. The methods described in [8] for solving the 2-dimensional case on a grid can easily be adapted to work for this more general version as well.

First we normalize the sets of rectangles, such that they lie on a d-dimensional grid, by replacing each vertex $p=(p_1, \cdots, p_d)$ by $R(p)=(R_1(p_1), \cdots, R_d(p_d))$ in the same way as for maximal elements. It is easily verified that two rectangles intersect if and only if their normalized versions intersect. For convenience, we assume that no two rectangles have a vertex with the same value in some coordinate.

The divide and conquer splits the universe into two equal halves with respect to the first coordinate. Hence, somewhere in the splitting process we are left with some portion $[A_1..B_1]$ on the first coordinate. During the process we maintain four sets of rectangles: V_1^i of rectangles in V_1 that lie completely in between A_1 and B_1 or intersect the left or right boundary of the range but not both, V_1^o of rectangles in V_1 that completely overlap $[A_1..B_1]$,and, similar, V_2^i and V_2^o for the rectangles in V_2. In the beginning V_1^i contains all rectangles in V_1, V_2^i contains all rectangles in V_2 and the other sets are empty.

Now assume we are left with a portion $[A_1..B_1]$. The rectangles in V_1^o intersect all rectangles in V_2^i and V_2^o with respect to the first coordinate. Hence, to find the intersecting pairs we solve the $d-1$ dimensional intersection problem with $V_1 = V_1^o$ and $V_2 = V_2^i \cup V_2^o$, excluding the first coordinate. Similar, we solve a $d-1$ dimensional intersection problem with $V_1=V_2^o$ and $V_2=V_1^i$. Next, we split the interval $[A_1..B_1]$ in two equal halves. For each half we build the four sets out of the rectangles in V_1^i and V_2^i. (The elements in V_1^o and V_2^o are no longer needed and can be discarded.) To guarantee low time bounds we normalize both halves in the same way

as described in Section 3. This reduces the universe in both halves to $(n/2)^d$. After this we solve the problem recursively in both halves. The splitting continues until the sets are empty or $B_1=A_1$.

Theorem 4.1. *The d-dimensional rectangle intersection problem can be solved in time $O(k + n\log^{d-1}n/\text{loglog } n)$, where k is the number of reported answers.*

Proof.

The first normalization step takes time $O(n \log n)$. Let $T(d,n)$ denote the amount of time required for solving the d-dimensional rectangle intersection problem on a grid of size n^d. We know

$$T(2,n) = O(k + \frac{n \log n}{\text{loglog } n})$$

Moreover, from the algorithm above it follows that

$$T(d,n) = 2T(d,n/2) + 2T(d-1,n) + O(n)$$

This immediately leads to

$$T(d,n) = O(k + \frac{n\log^{d-1}n}{\text{loglog } n})$$

□

Let us now concentrate on the boundary intersection problem. When two rectangles R_1 and R_2 boundary intersect each other, there must be some coordinate axis x_i such that, when projecting R_1 and R_2 on the hyperplane perpendicular to x_i, their projections they will boundary intersect. (Clearly, for each coordinate, the projections will intersect, but there might be only one direction in which their boundaries intersect.) This suggests the following algorithm.

We use the same method as described above. But rather that doing one divide and conquer with respect to the first coordinate, we repeat the method d times, once for each coordinate. This guarantees that, when solving a $d-1$ dimensional subproblem, we again only have to look for boundary intersections because for at least one coordinate the projections will boundary intersect. It is easy to verify that in this way all boundary intersections are reported.

Theorem 4.2. *The d-dimensional rectangle boundary intersection problem can be solved in time $O(k + n\log^{d-2}n \text{ loglog } n)$, where k is the number of reported answers.*

Proof.

The proof is similar to the case of true intersections. Let $T(d,n)$ denote the amount of time required for solving the d-dimensional rectangle intersection problem on a grid of size n^d. We know

$$T(2,n) = O(k + n \text{loglog } n)$$

Moreover, from the algorithm above it follows that

$$T(d,n) = d\ T'(d,n)$$

where

$$T'(d,n) = 2T'(d,n/2) + 2T(d-1,n) + O(n)$$

The time bound follows (because d is fixed).

□

5. Conclusions and open problems.

We have introduced a new technique, named *normalized divide and conquer* for solving a number of multi-dimensional problems more efficiently. The method first reduced the problems to problems on a multi-dimensional grid and next used a divide and conquer approach to reduce the size and dimension of the universe, resulting in low-dimensional versions of the problem that can be solved efficiently using known techniques. We successfully applied the method to the maximal elements problem and to some rectangle intersection problems.

We believe that many more multi-dimensional problems for which divide and conquer solutions exist can be solved more efficiently by using normalized divide and conquer. Problems like the measure problem, the closure problem and the connected components problem are investigated at the moment.

The time bounds obtained for the multi-dimensional problems highly depend on the efficiency of solutions on 2- and 3-dimensional grids. Improvements over the bounds obtained in [8] would immediately result in better multi-dimensional solutions. Especially the problem of finding all true intersection in a set of planar axis-parallel rectangles seems worth investigating. There is no indication that the bound obtained in [8] is optimal.

6. References.

[1] Bentley, J.L., Multidimensional divide and conquer, Comm. ACM 23 (1980), 214-229.

[2] Edelsbrunner, H., and M.H. Overmars, Batched dynamic solutions to decomposable searching problems, J. Algorithms 6 (1985), 515-542.

[3] Gabow, H.N., J.L. Bentley and R.E. Tarjan, Scaling and related techniques for geometry problems, Proc. 16th Annual ACM Symp. on Theory of Computing, 1984,135-143.

[4] Güting, R.H., Conquering contours, efficient algorithms for computational geometry, PhD thesis, Abteilung Informatik, Universität Dortmund, 1983.

[5] Güting, R.H., and D. Wood, Finding rectangle intersections by divide-and conquer, IEEE Trans. on Computers C-33 (1984), 671-675.

[6] Karlsson, R.G., Algorithms in a restricted universe, PhD thesis, Techn. Rep. CS-84-50, Dept. of Comp. Science, University of Waterloo, 1984.

[7] Karlsson, R.G., and J.I. Munro, Proximity on a grid, Proc. Second Symp. on Theoretical Aspects of Computer Science, Springer-Verlag Lect. Notes in Computer Science 182, 1985, 187-196.

[8] Karlsson, R.G., and M.H. Overmars, Scanline algorithms on a grid, Techn. Rep. RUU-CS-86-18, Dept of Comp. Science, University of Utrecht, 1986.

[9] Karlsson, R.G., and M.H. Overmars, Geometric algorithms on a grid, Techn. Rep., Dept. of Comp. Science, University of Utrecht, 1986. (to appear)

[10] Keil, J.M., and D.G. Kirkpatrick, Computational geometry on integer grids, Proc 19th Annual Allerton Conference, 1981, 41-50.

[11] Muller, H., Rastered point location, Proc. Workshop on Graphtheoretic Concepts in Computer Science (WG 85), Trauner Verlag, 1985, 281-293.

[12] Overmars, M.H., Range searching on a grid (ext. abstract), Proc. Workshop on Graphtheoretic Concepts in Computer Science (WG 85), Trauner Verlag, 1985, 295-305.

[13] Overmars, M.H., Efficient data structures for range searching on a grid, Techn. Rep., Dept. of Comp. Science, University of Utrecht, 1986. (to appear)

[14] Willard, D.E., Log-logarithmic worst-case range queries are possible in space $\Theta(n)$, Inform. Proc. Lett. 17 (1983), 81-84.

[15] Willard, D.E., New trie data structures which support very fast search operations, J. Comput. Syst. Sci. 28 (1984), 379-394.