Range Searching in Low-Density Environments*

Otfried Schwarzkopf† Jules Vleugels‡

Abstract

We define a set of arbitrarily-shaped objects in $\mathbb{R}^d$ to be a *low-density environment* if any axis-parallel hypercube intersects only few objects of comparable or larger size. Generalizing and simplifying previous results for fat objects, we present a data structure for point location in a low-density environment, and we show how this data structure can be extended to perform range searching queries with query ranges of size comparable to the smallest object.

Keywords: computational geometry, point location, range searching, fat objects, low density, motion planning.

1 Introduction

Many algorithms and data structures in computational geometry are considered "optimal" because there are examples of input configurations where the output to be constructed has size proportional to the running time of the algorithm. However, most constructions for such worst-case bounds are highly artificial, and, as less theoretically inclined researchers never fail to point out, "do not occur in practice."

The model of "fat objects" has been introduced to partially alleviate this problem. A fat object is one that has no long and skinny features—a simple example being a triangle whose angles are bounded from below by a given constant. Since most worst-case constructions involve arbitrarily long and thin objects, they cannot occur for sets of fat objects. It has therefore been argued that studying the behavior of an algorithm on a set of fat objects will give us a better approximation of its behavior on "real data."

The study of fat objects has recently received considerable attention in the computational geometry community. In particular, robotics applications have been reconsidered in this light. A comprehensive analysis of motion planning algorithms in the setting of fat objects has been given by van der Stappen in his thesis [8]. He could prove that some known algorithms perform much better than their theoretical worst-case bound in this setting—a result supporting the observations of practitioners—and he also gave some algorithms that have been tailored to exploit the fatness of objects.

*This research was partially supported by ESPRIT Basic Research Action No. 6546 (project PROMotion) and by the Netherlands' Organization for Scientific Research (NWO). The first author also acknowledges partial support by Pohang University of Science and Technology Grant P95015, 1995.

†Department of Computer Science, Pohang University of Science and Technology, Hyoja-dong, Pohang 790–784, South Korea. Email: otfried@postech.ac.kr

‡Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. Email: jules@cs.ruu.nl

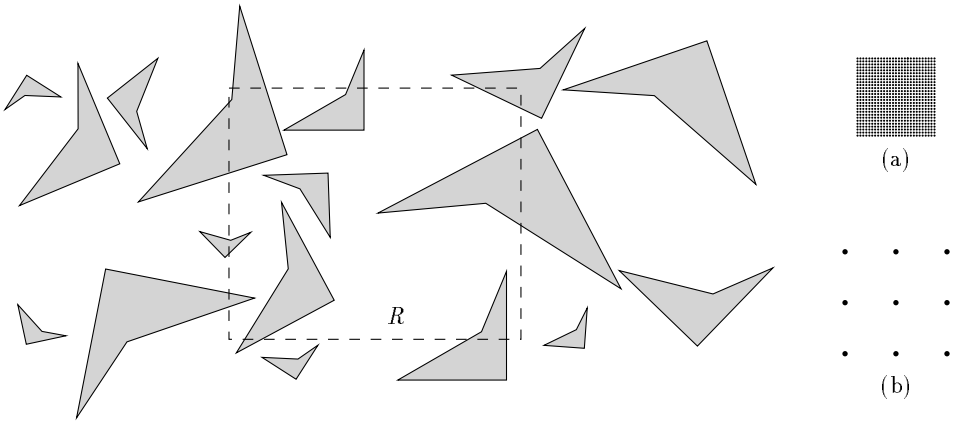

Figure 1: A set of 20-fat objects and a sample query range R.

Interestingly, most of the results in van der Stappen's thesis are based on a single property of fat objects, namely the fact that only few of them can intersect a region that is relatively small compared to the size of the objects. In this paper we argue that this property, which is in fact more general than the fatness condition, may be more suitable for the analysis of "real life" motion planning problems. The floorplan for a building, for instance, will often contain long and thin walls and is therefore not modelled well as a set of fat objects. Still, it may often form a *low-density environment* (a formal definition of this notion is given below).

The results of van der Stappen do not immediately generalize to this more general setting. The reason for this is that most of his algorithms are based on a data structure by Overmars and van der Stappen for range searching with small ranges [5]. This data structure stores a set of fat objects. A query consists of an arbitrarily shaped range whose size is comparable to the size of the smallest object, and returns the set of objects intersecting the range.

This data structure is interesting because it implements range searching queries by performing many point location queries in the set of objects, which are themselves implemented using a structure by Overmars [4]. The number of point location queries depends on the shape of the objects—van der Stappen could only prove bounds for convex and for polygonal fat objects—, their size, the size of the query range, and on the "fatness coefficient" of the objects (see below for details). Even in simple cases, this number can be quite large. In the example of Figure 1, for instance, 156,800 point location queries are necessary with their approach to determine the objects intersecting the query region R. (The two grids shown on the right hand side in the figure will be explained later.)

In this paper we show that the data structures for point location and for range searching with small ranges can indeed be generalized to low-density environments. As a consequence, most results of van der Stappen's thesis on robot motion planning generalize to low-density environments as well.

As a by-product, our approach is simpler than the previous one, works for arbitrarily shaped objects, and implements range searching using far fewer point location queries. For the example of Figure 1, for instance, our approach needs only 30 queries. Furthermore, our approach is robust in the sense that even if the objects do not satisfy the requirements for low-density, the algorithm will still deliver correct results. (Of course the complexity analysis will not hold anymore.) In contrast, the original approach is likely to deliver incorrect answers for inputs that are assumed to be t-fat for some $t > 0$ but are in fact only t'-fat for some $t' > t$.

This is particularly interesting because—as noted by van der Stappen [8]—computing the exact fatness coefficient of any given object is difficult. In our approach, we do not need to determine the actual parameters; they are used only in the analysis.

The data structure we suggest uses $O(n \log^{d-1} n)$ space and $O(n \log^d n)$ preprocessing time to store n objects in $\mathbb{R}^d$, such that point location queries and range queries with small ranges can be performed in time $O(\log^{d-1} n)$. We also give exact bounds that show the dependence on the size of the query range and on the low-density parameter.

2 Fatness and low-density environments

As mentioned in the introduction, the concept of fatness has received quite some attention recently [1, 2, 3, 4, 9, 10]. For completeness, we review the definition of fatness used by Overmars and van der Stappen [5]. They define an object E in $\mathbb{R}^d$ to be t-fat, for a given parameter $t > 0$, if for every axis-parallel hypercube H with center in E that does not contain E fully, the volume of $H \cap E$ is at least $1/t$ of the volume of H. This definition forbids a fat object to be long and thin or to have long and thin parts. As a result, only few disjoint fat objects can intersect a region of size comparable to the smallest object. We argue that this property should be used in the analysis instead of the fatness itself, since it seems to hold for more real-world settings. (Related but somewhat different concepts that result in a relatively low object density are *bounded local complexity* studied by Schwartz and Sharir [7] and *dispersion* studied by Pignon [6].)

In the following, a *box* is an axis-parallel hypercube. The *size of a box* is the length of any of its sides. Given an object $E \subset \mathbb{R}^d$, we define $\rho(E)$ to be the size of a smallest box enclosing E, and we call $\rho(E)$ the *size* of E. An object E is smaller than an object E' if the size of E is smaller than the size of E'.

We now define a *low-density environment*:

Definition 2.0.1 *Given a set $\mathcal{E}$ of (not necessarily disjoint) objects and a number $k > 0$, we call $\mathcal{E}$ a* k-low-density environment *if for every box H we have*

$$|\{E \in \mathcal{E} \mid E \cap H \neq \emptyset \wedge \rho(E) \geqslant \rho(H)\}| \leqslant k.$$

In other words, any box cannot intersect more than a constant number of objects of the same or larger size. Note that the definition implies that a box H intersects at most a constant number of objects E of size $\rho(E) \geqslant c\rho(H)$ when c is a constant, since such a box can be covered by a constant number of boxes of size $c\rho(H)$.

As mentioned above, van der Stappen [8] observed that any set of disjoint fat objects forms a low-density environment. To be more precise, a set $\mathcal{E}$ of disjoint t-fat objects in $\mathbb{R}^d$ is a $2^d t$-low-density environment. On the other hand, not every low-density environment consist of fat objects. Consider a set consisting of objects identical to the one shown in Figure 2. The objects are not t-fat for any $t > 0$ because of the thin protruding edge. However, it is easily seen that at most a constant number of such objects can intersect a region with minimal enclosing hypercube size at most $\rho(E)$.

Our data structure can be used for objects and ranges of arbitrary shape. We only need a small set of operations, and for sake of simplicity we will assume that we can do all of them in constant time:

- Given an object, find a smallest enclosing box for the object

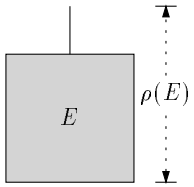

Figure 2: An object that is not t-fat for any $t > 0$.

- Given an object and a point, box, or range, test whether they intersect
- Given a range and a box, determine whether they intersect

This is a natural assumption if both objects and ranges have constant description complexity.

To ensure that we do not generate large hidden constants, we will carry all parameters through the analysis, and will only assume that the dimension of the space is constant.

3 Point location in low-density environments

In this section we briefly describe a generalization of the structure for point location introduced by Overmars [4] to low-density environments. We are given a k-low-density environment $\mathcal{E}$, and we want to store it such that given a query point $q \in \mathbb{R}^d$ we can quickly decide which objects $E \subseteq \mathcal{E}$, if any, contain q.

Given $E \in \mathcal{E}$, we choose a smallest enclosing box $C(E)$ for E. By definition, we have $\rho(C(E)) = \rho(E)$. We can now define $\mathcal{E}(E)$ as the set of objects that are at least as large as E and that intersect its enclosing box $C(E)$:

$$\mathcal{E}(E) = \{E' \mid E' \cap C(E) \neq \emptyset \wedge \rho(E') \geqslant \rho(E)\}$$

It follows directly from our definition of low-density environment that $|\mathcal{E}(E)| \leqslant k$. The following easy lemma by Overmars [4] is the crucial argument for the correctness of his point location structure:

Lemma 3.0.1 *[4] Given a point q. Let E be the smallest $E \in \mathcal{E}$ with $q \in C(E)$. If $q \in E'$ for some $E' \in \mathcal{E}$, then $E' \in \mathcal{E}(E)$.*

This suggests that we can find the answer to a point location query with a point q as follows. If no hypercube $C(E)$ contains q, then q clearly lies in none of the objects. Otherwise, let $C(E)$ be the smallest hypercube containing q; by Lemma 3.0.1 we know that all objects containing q must be present in $\mathcal{E}(E)$. We can therefore test in $O(k)$ time which objects actually contain q.

Overmars [4] uses a multi-level segment tree to store the set of boxes $C(E)$. This structure needs $O(n \log^{d-1} n)$ space and preprocessing time, and it allows us to find in query time $O(\log^{d-1} n)$ a smallest size box containing a query point q.

With every box $C(E)$ we have to store the set $\mathcal{E}(E)$ of size at most k, so the total space requirement of the structure is $O(n \log^{d-1} n + nk)$. We will see later how to compute the sets $\mathcal{E}(E)$ efficiently, and state the following theorem.

Theorem 3.1 *A k-low-density environment $\mathcal{E}$ of n objects in $\mathbb{R}^d$ can be stored in a data structure of size $O(n \log^{d-1} n + kn)$, such that it takes $O(\log^{d-1} n + k)$ time to report all objects $E \in \mathcal{E}$ that contain a given query point $q \in \mathbb{R}^d$. The data structure can be computed in time $O(n \log^d n + nk \log n)$.*

4 Range searching for small ranges

We consider the following problem: We are given a k-low-density environment $\mathcal{E}$ consisting of a set of n possibly intersecting objects in $\mathbb{R}^d$. A query consists of an arbitrarily-shaped region R, and we would like to report all objects $E \in \mathcal{E}$ that intersect R.

We let $h := \rho(R)/\rho_0$, where ρ_0 denotes the minimum of $\rho(E)$, for $E \in \mathcal{E}$. In the original setting by Overmars and van der Stappen, h was considered a constant; we will consider it a parameter.

Since $\mathcal{E}$ is a low-density environment, every hypercube H with $\rho(H) = \rho_0$ intersects at most k objects. Since R can be covered by at most h^d such hypercubes, the answer to such a query consists of at most $h^d k$ objects.

Overmars and van der Stappen have shown [5] that a range search query can be solved for a set of t-fat disjoint convex objects by answering a large number of point location queries arranged as a regular orthogonal grid with resolution (that is, distance between adjacent grid points) $2\rho_0/(td^{d+1/2})$. That implies that $((2h+4)/(td^{d+1/2}))^d$ point location queries are sufficient to answer a range query. (They also prove a similar bound for a set of polygonal fat objects.) Although this is $O(1)$ if both d and h are constant, in practical situations it may become unfeasibly large: a range query among convex objects with moderate values of $t = 10$, $d = 3$, and $h = 10$ would already require more than 10^{10} point location queries.

We will use a different way to translate the range query into a set of point location queries, using a grid of much coarser resolution. To ensure that we still correctly report all objects intersected by the query region, we only need to expand all the objects somewhat.

Let $H(\alpha)$ denote the hypercube of size α centered at the origin, that is

$$H(\alpha) := \{(x_1, \ldots, x_d) \mid -\alpha/2 \leqslant x_i \leqslant \alpha/2\},$$

and define the *expansion* $\Pi(X, \alpha)$ of a set $X \subset \mathbb{R}^d$ by distance $\alpha \geqslant 0$ as the Minkowski sum $X + H(\alpha)$. As a special case, $\Pi(p, \alpha)$ denotes the box of size α centered at the point p.

Given an object $E \in \mathcal{E}$, we define the expanded enclosing box as $C^*(E) := \Pi(C(E), 2\rho_0)$. As before, we can define the set $\mathcal{E}^*(E)$ of objects that are at least as large as E and that intersect the expanded enclosing box $C^*(E)$:

$$\mathcal{E}^*(E) = \{E' \mid E' \cap C^*(E) \neq \emptyset \wedge \rho(E') \geqslant \rho(E)\}$$

We observe first that $\mathcal{C}^*(E)$ can be covered by 2^d boxes of size $\rho(E)$, and hence $|\mathcal{E}^*(E)| \leqslant 2^d k$. The critical lemma for range searching is the following

Lemma 4.0.1 *Given a point p and a box $H = \Pi(p, \rho_0)$ of size ρ_0 centered at p. Let E be the smallest $E \in \mathcal{E}$ with $p \in C^*(E)$. If $H \cap E' \neq \emptyset$ for some $E' \in \mathcal{E}$, then $E' \in \mathcal{E}(E)$.*

Proof. Since $E' \cap H \neq \emptyset$, we have $p \in \Pi(E', \rho_0)$, and therefore $p \in C^*(E')$. Consequently, $\rho(E) \leqslant \rho(E')$.

On the other hand, $p \in C^*(E)$ implies that $\Pi(E, \rho_0)$ and $\Pi(E', \rho_0)$ intersect. Therefore, $E' \cap \Pi(E, 2\rho_0) \neq \emptyset$, and since $\rho(E') \geqslant \rho(E)$, we have $E' \in \mathcal{E}^*(E)$. □

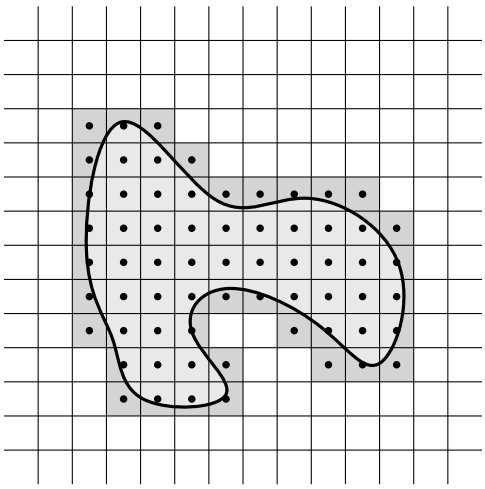

Figure 3: Selecting the sample points for a range R.

This lemma is again the basis for our data structure. As in the previous section, we build a multi-level segment tree to store the set of boxes $\{C^*(E) \mid E \in \mathcal{E}\}$, such that we can find a smallest box containing a query point q. This takes storage and preprocessing time $O(n \log^{d-1} n)$, the query time is $O(\log^{d-1} n)$. With every box $C^*(E)$ we store the set $\mathcal{E}^*(E)$, taking $O(nk)$ space in total. (Again we postpone the discussion of how to find the sets $\mathcal{E}^*$ to a later section.)

Consider now a grid $\mathcal{G}$ of resolution ρ_0, so $\mathcal{G} = \{(a_1\rho_0, a_2\rho_0, \ldots, a_d\rho_0) \mid a_i \in \mathbb{Z}\}$. For a grid point $p \in \mathcal{G}$, let $G(p)$ denote the grid cell of p, $G(p) = \Pi(p, \rho_0)$.

To answer a range query with a range R, we determine the set $\mathcal{G}(R)$ of grid points p with $G(p) \cap R \neq \emptyset$, see Figure 3. Since the size of R is at most $h\rho_0$, there are at most h^d such grid points p. For every point $p \in \mathcal{G}(R)$, we determine in $O(\log^{d-1} n)$ time a smallest box $C^*(E(p))$ containing p. By Lemma 4.0.1, any object E' that intersects $G(p)$ must be contained in $\mathcal{E}^*(E(p))$. We collect the set of all objects appearing in one of the $\mathcal{E}^*(E(p))$, for $p \in \mathcal{G}(R)$. By the above, this set has cardinality at most $h^d 2^d k$, and it contains all objects intersecting R. As the final step, we only have to check for every object whether it actually intersects R. The total query time is $O(h^d \log^{d-1} n + h^d k)$.

In anticipation of the result of Section 6 that describes the construction of the data structure, we summarize the main result of this section in the following theorem.

Theorem 4.1 *Let $\mathcal{E}$ be a k-low-density environment in $\mathbb{R}^d$. The set $\mathcal{E}$ can be stored in a data structure of size $O(n \log^{d-1} n + kn)$, such that a range searching query with a region $R \subset \mathbb{R}^d$ of size at most $h \cdot \rho_0$ among $\mathcal{E}$ can be answered in time $O(h^d \log^{d-1} n + h^d k)$. This data structure can be built in time $O(n \log^d n + nk \log n)$.*

5 Practical complexity: an example

Although the data structure suggested here has the same asymptotic storage and query time as the structure by Overmars and van der Stappen, we expect it to perform better in practice. In this paper we have not assumed that k and h are constant, to show that their contribution

is quite moderate and that there are no hidden constants of astronomical size. Let's illustrate this with an example.

We consider the set of objects shown earlier in Figure 1. The objects shown here are 20-fat [8], so we can apply the range searching data structure by Overmars and van der Stappen. By plugging in the values for the diameter of the query region, the fatness and combinatorial complexity of the objects into their formula, we find that the required grid resolution is 0.00625. (A grid at *one tenth* of this resolution is shown in Figure 1a.) The number of point locations required by their analysis to answer this particular query is bounded by $156,800$. Clearly this would not be very useful in a practical application.

Our method, on the other hand, uses a grid of resolution ρ_0, shown in Figure 1 at the bottom right. The number of point location queries needed is at most 30.

6 Building the data structure

So far we have avoided the issue of computing the sets $\mathcal{E}(E)$ and $\mathcal{E}^*(E)$ used in the data structures we described. While the multi-level data structure to store the enclosing hypercubes $C(E)$ can be built in $O(n \log^{d-1} n)$ time using standard techniques, the computation of the sets $\mathcal{E}(E)$ is surprisingly difficult, and was actually left unresolved in the original paper [4]. Overmars and van der Stappen [5] showed how to compute these sets by using their technique for range searching itself, adding the objects in order of decreasing size. Using dynamic fractional cascading, their algorithm runs in time $O(n \log^{d-1} n \log\log n)$ for convex or polygonal objects.

We here describe a simple approach using a divide-and-conquer algorithm, running in time $O(n \log^d n + nk \log n)$. We do not know whether dynamic fractional cascading could be employed to reduce it to the bound of Overmars and van der Stappen.

In the following, we are given a k-low-density environment $\mathcal{E}$, and we want to compute the sets $\mathcal{E}^*(E)$, for $E \in \mathcal{E}$. Note that once we have found $\mathcal{E}^*(E)$, we can easily determine $\mathcal{E}(E) \subseteq \mathcal{E}^*(E)$ in total time $O(nk)$.

We start by splitting the set of objects $\mathcal{E}$ into two sets $\mathcal{E}_0$ of the $n/2$ smaller objects and $\mathcal{E}_1$ of larger objects. In other words, $\rho(E_0) \leqslant \rho(E_1)$ for any $E_0 \in \mathcal{E}_0$, $E_1 \in \mathcal{E}_1$. We recursively compute the data structure of Section 4 for $\mathcal{E}_0$ and $\mathcal{E}_1$.

Note that as a result we will get for every $E \in \mathcal{E}_i$, where $i = 0$ or 1, the set

$$\mathcal{E}_i^*(E) = \{E' \in \mathcal{E}_i \mid E' \cap \Pi(C(E), 2\rho_i) \neq \emptyset \wedge \rho(E') \geqslant \rho(E)\},$$

where $\rho_i = \min_{E \in \mathcal{E}_i} \rho(E)$. (Of course, ρ_0 is also the minimum $\rho(E)$ over all $E \in \mathcal{E}$, as we defined it earlier.)

It remains to show how to compute the data structure for $\mathcal{E}$ from the data structures for $\mathcal{E}_0$ and $\mathcal{E}_1$. As mentioned before, the multi-level segment tree for $\mathcal{E}$ can be constructed from scratch in time $O(n \log^{d-1} n)$, so we only have to describe how to find the sets $\mathcal{E}^*(E)$, for every $E \in \mathcal{E}$.

Consider first the case $E \in \mathcal{E}_1$. Since $\rho_0 \leqslant \rho_1$, we have $\mathcal{E}^*(E) \subseteq \mathcal{E}_1^*(E)$, and we can easily determine $\mathcal{E}^*(E)$ in time $O(k)$.

So assume that $E \in \mathcal{E}_0$. Clearly,

$$\mathcal{E}^*(E) = \mathcal{E}_0^*(E) \cup \{E' \in E_1 \mid E' \cap \Pi(C(E), 2\rho_0) \neq \emptyset\}$$

This implies that we can determine $\mathcal{E}^*(E)$ by doing a range searching query with $\Pi(C(E), 2\rho_0)$ using the range searching data structure for $\mathcal{E}_1$. By Theorem 4.1, this takes $O(\log^{d-1} n + k)$ time.

We get the following recursion for the preprocessing time $T(n)$:

$$T(n) = 2T(n/2) + O(n \log^{d-1} n + nk),$$

which solves to $O(n \log^d n + nk \log n)$.

References

[1] A. Efrat, M. Sharir, and G. Rote. On the union of fat wedges and separating a collection of segments by a line. *Comput. Geom. Theory Appl.*, 3:277–288, 1994.

[2] M. J. Katz, M. Overmars, and M. Sharir. Efficient output sensitive hidden surface removal for objects with small union size. *Comput. Geom. Theory Appl.*, 2:223–234, 1992.

[3] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl. Fat triangles determine linearly many holes. *SIAM J. Comput.*, 23:154–169, 1994.

[4] M. H. Overmars. Point location in fat subdivisions. *Inform. Process. Lett.*, 44:261–265, 1992.

[5] M. H. Overmars and A. F. van der Stappen. Range searching and point location among fat objects. In J. van Leeuwen, editor, *Algorithms – ESA '94*, volume 855 of *Lecture Notes in Computer Science*, pages 240–253, Utrecht, NL, September 1994.

[6] Ph. Pignon. *Structuration de l'Espace pour une Planification Hiérarchisée des Trajectoires de Robots Mobiles.* Ph.D. Thesis, LAAS-CNRS and Université Paul Sabatier de Toulouse, Toulouse, France, 1993.

[7] J. T. Schwartz and M. Sharir. Efficient motion planning algorithms in environments of bounded local complexity. Report 164, Dept. Comput. Sci., Courant Inst. Math. Sci., New York Univ., New York, NY, 1985.

[8] A. F. van der Stappen. *Motion planning amidst fat obstacles.* Ph.D. Thesis, Dept. Comput. Sci., Utrecht Univ., Utrecht, the Netherlands, 1994.

[9] A. F. van der Stappen, D. Halperin, and M. H. Overmars. The complexity of the free space for a robot moving amidst fat obstacles. *Comput. Geom. Theory Appl.*, 3:353–373, 1993.

[10] Marc van Kreveld. On fat partitioning, fat covering, and the union size of polygons. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes in Computer Science*, pages 452–463. Springer-Verlag, 1993.