Algorithms for Two-Box Covering

Esther M. Arkin
Dept. of Applied Mathematics and Statistics
Stony Brook University
Stony Brook, NY 11794

Gill Barequet
Dept. of Computer Science
The Technion—Israel Institute of Technology
Haifa 32000, Israel

Joseph S. B. Mitchell
Dept. of Applied Mathematics and Statistics
Stony Brook University
Stony Brook, NY 11794

ABSTRACT

We study the problem of covering a set of points or polyhedra in $\Re^3$ with two axis-aligned boxes in order to minimize a function of the measures of the two boxes, such as the sum or the maximum of their volumes. This *2-box cover* problem arises naturally in the construction of bounding volume hierarchies, as well as in shape approximation and clustering. Existing algorithms solve the min-max version of the exact problem in quadratic time. Our results are more general, addressing min-max, min-sum and other versions. Our results give the first approximation schemes for the problem, which run in nearly linear time, as well as some new exact algorithms. We give $(1+\varepsilon)$-approximation algorithms for minimizing the maximum or sum of volumes (or surface areas, diameters, widths, or girths) of the two boxes in $\Re^3$. We investigate also the problem of computing balanced coverings, in which each box covers at least a fraction of the input objects, and we discuss the application to constructing provably-good bounding volume hierarchies of polyhedra. We also generalize our results to higher dimensions.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*combinatorial algorithms*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms

Algorithms

Keywords

Exact and approximate algorithms

1. INTRODUCTION

A common method of preprocessing spatial data for efficient manipulation and queries is to construct a spatial hierarchy, such as a bounding volume tree (BV-tree). Various types of BV-trees have been widely used in computer graphics and animation (e.g., for collision detection, visibility culling, and nearest neighbor searching) and in spatial databases (e.g., for GIS and data mining). In order to be maximally effective at allowing pruning during a search of the hierarchy, it is important that the bounding volumes fit the data as tightly as possible. In a top-down construction of the BV-tree, each step requires that one compute a good way to partition the data corresponding to a node of the hierarchy so that the data corresponding to the children of the node have bounding volumes that are as "small" as possible.

For the commonly used bounding volume of axis-aligned bounding boxes (AABBs), we refer to the corresponding optimization problem as the *2-box cover problem.*

More generally, let S be a set of n points (or polyhedra) in $\Re^d$. Our goal is to find a partition of S into two sets, S_1 and S_2, in order to minimize the *combined measure*, $f(B_1, B_2)$, of the axis-aligned bounding boxes, B_1 and B_2, of the subsets S_1 and S_2.

We denote by $(x(B), y(B), z(B))$ the lengths of the coordinate extents of an axis-aligned box $B \subset \Re^3$. We consider various ways of quantifying the "size" of B. A *measure*, $\mu(B)$, of B is a nonnegative function that obeys $\mu(B) \le \mu(B')$ if $B \subseteq B'$. In much of our discussions we will focus on the *volume* measure, given by $\mu(B) = x(B) \cdot y(B) \cdot z(B)$, but most of our results apply also to the measures of *surface area* $(x(B)y(B)+x(B)z(B)+y(B)z(B))$, *diameter* (in L_∞) $(\max(x(B), y(B), z(B)))$, *width* $(\min(x(B), y(B), z(B)))$, and *girth* $(x(B)+y(B)+z(B))$.

By the *combined measure* of two boxes we mean some function $f(B_1, B_2)$ that satisfies monotonicity: if $B'_1 \subseteq B_1$ and $B'_2 \subseteq B_2$, then $f(B'_1, B'_2) \le f(B_1, B_2)$. Often, we focus on the *min-max* $(f(B_1, B_2) = \max\{\mu(B_1), \mu(B_2)\}$, for some measure μ), the *min-sum* $(f(B_1, B_2) = \mu(B_1) + \mu(B_2))$, or the *union* $(f(B_1, B_2) = \mu(B_1 \cup B_2))$ combined measure, but some of our results apply more generally to other combined measures.

1.1 Summary of Results

We have obtained the following results:

(1) In $\Re^3$, a $(1+\varepsilon)$-approximation of the min-max 2-box cover can be computed in time $O(\min\{n \log n, n/\varepsilon\} + (\log n)/\varepsilon^2)$. A $(1+\varepsilon)$-approximation of the min-sum 2-box cover can be computed in time $O(\min\{n \log n, n/\varepsilon\} + n/\varepsilon^2)$.

(2) The exact min-sum (or min-union) 2-box cover can be computed in time $O(n^d)$ in $\Re^d$.

(3) The min-sum (resp., min-max) surface area (or $(d-1)$-volume) for a 2-box cover of $S \subset \Re^d$ can be $(1+\varepsilon)$-approximated in time $O(n+(1/\varepsilon)^d)$ (resp., $O(n+(1/\varepsilon)^{d-1})$).

(4) We extend the min-sum and min-max surface area results to obtain $(1+\varepsilon)$-approximation algorithms for 2-box covers that obey a balance (cardinality) constraint, in that each box contains at least a certain fraction, α, of the elements of S.

(5) We show how to apply our methods to covering sets of polygons or polyhedra. We know of no prior result giving a provably-good covering of a polyhedron by two AABBs. (Typically, in graphics applications, simple heuristics are used to partition the facets of a polyhedron; however, each of the standard methods can perform arbitrarily poorly in the worst case.) Our approximation algorithms also then give corresponding provable results on the quality of bounding volume hierarchies.

1.2 Related Work

Bespamyatnikh and Segal [8] have solved the exact min-max 2-box cover problem for point sets in $\Re^d$ with an algorithm that takes time $O(n \log n + n^{d-1})$ and space $O(n)$. Their algorithm minimizes the maximum of the measures of the boxes, for any monotone measure function (e.g., volume, surface area, diameter, etc). Their algorithm crucially uses the structure of the min-max case and does not apply, e.g., to the min-sum problem. Here, we consider min-max, min-sum, or, more generally, any combined measure for the two boxes.

There is a large family of related clustering problems, which include the *k-center* problem (minimize the maximum radius in a covering with k disks), the *k-median* problem (minimize the sum of distances to the centers), and the *k-clustering* problem (minimize the sum of all interpoint distances within a cluster). Most facility-location problems are hard (NP-complete) if k is part of the input. The Euclidean k-medians problem has a PTAS [4]. Recent applications of core-sets have yielded efficient approximation schemes for 2-centers and k-centers in higher dimensions (see Bădoiu et al. [10] and Kumar et al. [27]). For a survey of results for the Euclidean instances of clustering problems, see Agarwal and Sharir [2].

The min-max 2-box cover is related to the rectilinear 2-center problem (cover by two axis-aligned cubes of size r, for the smallest value of r). This problem has been solved in linear time [12, 33], e.g., using the fact that it is an LP-type problem. Our problem is substantially different from the 2-center problem, in that we have considerably more degrees of freedom in covering by boxes, which may be very far from squares, and may be of very different sizes.

The min-sum 2-box cover is most closely related to the *min-size k-clustering* of Bilò et al. [9] (see also Lev-Tov and Peleg [28] and Alt et al. [3]), in which the goal is to minimize the sum of the radii of the disks that cover a set of points. In the plane, the exact solution of min-size clustering for $k=2$ has a slightly-subquadratic time algorithm ($O(n^2/\log\log n)$) due to Hershberger [20]. Charikar and Panigrahy [11] give an $O(1)$-approximation algorithm in metric spaces for minimizing the sum of radii using at most k clusters.

1.3 Motivation and Applications

The 2-box cover of a set is a fundamental problem that shows up in shape approximation, bounding volume hierarchies, intersection detection, and range-search data structures (such as R-trees and their numerous variants [16, 1]).

There has been considerable research on the problem of collision detection in simulation systems, where bounding volume hierarchies are commonly used. For surveys, see, e.g., [23, 29]. Of most relevance to our work here is the widely-used method of bounding volume hierarchies (BV-trees). The choice of bounding volume was often spheres [21, 22, 30] or axis-aligned bounding boxes (AABBs) [7, 15, 19], due to the simplicity in checking two such volumes for overlap (intersection) and in transforming AABBs when objects move. The system *SOLID* [34, 35, 36] achieves its efficiency using a carefully engineered hierarchy of AABBs. Another popular bounding volume is the oriented bounding box (OBB), a bounding box of arbitrary orientation, which is used in BOXTREEs [6] and OBBTrees [14], the basis for the popular *RAPID* system. The *QuickCD* system [18, 25] is based on BV-tree methods utilizing bounding "k-dops" (discrete orientation polytopes), whose supporting hyperplanes have normals from a small number, k, of selected orientations [24]. Zachmann [39, 40] and He [17] have developed related approaches ("DOP-trees," "restricted boxtrees," and "QuOSPS trees"). Other shapes which have been used in hierarchical decompositions are truncated tetrahedra [31], triangular prisms [6], cones [32], ellipsoids [ibid.], etc. BV-trees are also useful for ray tracing [5, 13, 24, 37] and for proximity queries [26, 38].

2. PRELIMINARIES

We let $x_i(B)$ denote the *extent* (length) of an axis-aligned box B in the x_i-coordinate direction. We let BB(X) denote the axis-aligned bounding box of X. Without loss of generality, we assume that BB$(S) = U$, the unit cube $[0,1]^d$.

Since the MAX-GAP problem (the min-sum 2-interval cover in one dimension) has an $\Omega(n \log n)$ lower bound, we get a similar result for exactly solving the 2-box cover according to various measures in three dimensions:

PROPOSITION 1. *The min-sum 2-box cover in $\Re^3$, for measures volume, surface area, diameter, width, and girth, requires $\Omega(n \log n)$ time in the worst case.*

PROOF. The proof is based on a reduction from the MAX-GAP problem to our problem. Given an unordered set $X = \{x_1, x_2, \ldots, x_n\} \subset \Re^1$ of n numbers, which is an input to MAX-GAP, we map X to the set $S = \{(x_i, 0, 0) : x_i \in X\} \cup \{(x_{min}, 1, 1), (x_{max}, 1, 1)\}$ of $n+2$ points in $\Re^3$, where $x_{min} = \min_i x_i$ and $x_{max} = \max_i x_i$. It is easy to see that an optimal solution to the min-sum problem for any of the measures volume, surface area, diameter, or girth is to cover the points of S with two boxes whose x-extents cover the interval $[x_{min}, x_{max}]$, except for exactly the interval corresponding to a maximum gap.

For the case of width measure, we proceed similarly, mapping X to a slightly different set $S = \{(x_i, 0, 0) : x_i \in X\} \cup \{(x_{min}, 1, 1), (x_{max}, -1, -1)\}$. Then, an optimal solution to the min-sum problem is to partition S with a separating plane orthogonal to the x-axis at the maximum gap. $\square$

Remark: It is an interesting open problem if the exact min-max 2-box cover has an $\Omega(n \log n)$ lower bound, in two or more dimensions.

We solve the min-sum problem (and other versions) exactly, using the fact that many of the facets of the optimal boxes are determined by the facets of the bounding box, $\mathrm{BB}(S)$:

THEOREM 2. *The min-sum 2-box cover problem (with respect to any monotone measure) in 3D can be solved in $O(n^3)$ time exactly. In d dimensions, it can be solved in $O(dn^d)$ time exactly. These results apply to any combined measure for which one can update the measure in constant time upon the addition or deletion of a point.*

PROOF. (Sketch) The bounding box $\mathrm{BB}(S)$ has $2d$ facets, each of which must be coplanar with at least one facet of one of the two boxes in an optimal solution. Thus, one box, say B_1, must have at least d of its $2d$ facets coplanar with facets of $\mathrm{BB}(S)$. This leaves at most d facets of B_1 still to specify, which we can do by enumerating all possibilities of which points of S determine which unspecified facets of B_1. With the points of S presorted according to each coordinate, it is not hard to enumerate all $O(n^d)$ possibilities at a cost of $O(d)$ per choice. $\square$

3. A SIMPLE GRID-BASED APPROXIMATION

We now discuss a simple grid-based method of approximating the optimal solution OPT. Consider a regular grid of $k \times k \times k$ voxels within $\mathrm{BB}(S) = U \subset \Re^3$. The first step of the algorithm is to bucket the points in S into the grid, in time $O(n)$, obtaining the set of *occupied* voxels. The number, N, of occupied voxels satisfies $N \leq \min\{n, k^3\}$. After noting that the occupied voxels are already sorted, a direct application of the exact methods for min-max [8] and min-sum (Theorem 2) yield that we can solve the min-max 2-box cover of the occupied voxels in time $O(n + N^2) = O(n + (1/\varepsilon)^6)$ and solve the min-sum 2-box cover of the occupied voxels in time $O(n + N^3) = O(n + (1/\varepsilon)^9)$. These bounds can be improved, though, by noticing that the exact algorithms depend not on the number of points, per se, but rather on the number of distinct x-, y-, and z-coordinates. In the grid, these numbers are just k (not N), so we get the bounds in the following proposition, which we state in the more general d-dimensional setting:

PROPOSITION 3. *An optimal grid-based solution to the min-sum 2-box cover problem in $\Re^d$ can be computed in time $O(n + k^d)$, where k is the discretization resolution of each coordinate. For the min-max version, the time bound is $O(n + k^{d-1})$. (These bounds assume use of the floor function.)*

How good is the grid-based 2-box cover solution relative to OPT? In general, with respect to optimizing volume and considering multiplicative approximation factors, the answer is that the solution can be arbitrarily bad. In Figure 1 we show an example for which the simple grid-based solution is an arbitrarily-large factor greater than OPT for area minimization in 2D (and it applies also to volume minimization in 3D), even if the grid-based solution is post-processed to shrink the boxes around the points of S they contain.

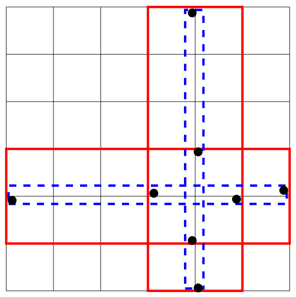

Figure 1: The area/volume-optimizing simple grid-based solution gives the two red boxes (both for min-max and for min-sum), while the optimal solution (shown in blue, dashed) is smaller by an arbitrarily-large factor.

Let us consider the analysis more carefully. Each of the two boxes, B_1 and B_2, of OPT can be rounded out to the discrete coordinates of the grid; this results in adding at most $1/k$ to each extent of a box. For an extent that is initially "large" (meaning $\Omega(1/k)$), this rounding causes only a constant factor increase in the extent. For a small extent, though, the rounding out can increase the extent by an unbounded factor, causing any measure that depends on that extent (e.g., volume) to increase by a large factor too.

One case for which the grid-based solution gives a provable bound on the approximation factor is that of minimizing surface area. (We focus here on the 3D case and speak of surface area, but the result lifts to d dimensions when we are concerned with measures based on non-full-dimensional measures, such as the $(d-1)$-dimensional volume of the facets of the boxes.)

First, we observe that if B_1 and B_2 are not separable, then their surface area is "substantial" (at least 2):

LEMMA 4. *For any two axis-aligned boxes B_1 and B_2 that cover S and that are not disjoint, the sum of the surface areas of B_1 and B_2 is at least 2, and, thus, the larger of the two surface areas is at least 1.*

PROOF. Let us distinguish between the four possible non-separable cases, as in Figure 2(b-e). In cases (b) and (e), we scan U along the x axis (from left to right), sweeping with a plane orthogonal to the x-axis. At every intermediate $0 \leq x \leq 1$, at least one box is active, and its yz cross-section must span either the entire y or z dimension of U. Thus, the perimeter of the yz cross-section of the active box is at least 2. Integrating this along the x dimension, we obtain the lower bound. In case (d) the surface area is minimized when the intersection of the two boxes is exactly a point, in which case the combined surface area, $\sum_{i=1}^{2}(x(B_i)y(B_i) + x(B_i)z(B_i) + y(B_i)z(B_i))$, is subject to $\sum_{i=1}^{2} x(B_i) = \sum_{i=1}^{2} y(B_i) = \sum_{i=1}^{2} z(B_i) = 1$. Elementary calculus shows that the minimum is obtained when $x(B_i) = y(B_i) = z(B_i) = 0.5$ (for $i = 1, 2$), in which case the total surface area of the two boxes is 3. In case (c), we can shift down the left box (without changing its surface area) along the x-axis until it touches the bottom of U, then lift up the bottom face of the right box until in aligns with the top face of the left box. This only reduces the total surface area, but now we are in case (d). $\square$

If a box B having extents $(x(B), y(B), z(B))$ is rounded out, with each extent increasing by at most $\Delta = 1/k$, then the surface area goes up by at most

$$4\Delta(x(B) + y(B) + z(B)) + 6\Delta^2 \leq 6\Delta(2 + \Delta) \leq 13\Delta,$$

assuming $k \geq 6$ for the last inequality.

Consider an optimal min-sum 2-box cover given by two boxes B_1 and B_2.

Case (1). If B_1 and B_2 intersect, then by Lemma 4 we know that the sum of their surfaces areas is at least 2. The simple grid-based solution has the option to output the rounded out versions, $\bar{B}_1$ and $\bar{B}_2$, of B_1 and B_2; it may output a different pair of grid-based boxes, but the sum of the surface areas of the boxes it outputs is at most that of $\bar{B}_1$ and $\bar{B}_2$. Now, the sum of the surface areas of $\bar{B}_1$ and $\bar{B}_2$ is at most $OPT + 2(13\Delta) \leq (1 + \varepsilon)OPT$, if we pick $k \geq 13/\varepsilon$.

Case (2). If B_1 and B_2 are disjoint, then there are two subcases. If the rounded out boxes $\bar{B}_1$ and $\bar{B}_2$ are not disjoint (they overlap or touch each other), then we know that the sum of the surface areas of $\bar{B}_1$ and $\bar{B}_2$ is at most $OPT + 2(13\Delta) \leq (1 + \varepsilon)OPT$, if we pick $k \geq 13/\varepsilon$, and the simple grid-based solution gives us a pair of boxes with at most this surface area (in sum). If the rounded out boxes $\bar{B}_1$ and $\bar{B}_2$ are disjoint, we can solve this case by using the exact algorithm; the sorting step can be omitted since the buckets are already sorted. We can then shrink the grid-based boxes to output the minimum enclosing boxes of each of the two subsets of S to yield an optimal (separable) solution.

A similar argument holds for the min-max case. Further, the arguments lift to d dimensions in addressing the surface area ($(d-1)$-volume) measure. In summary, we have sketched the proof of the following:

THEOREM 5. *The min-sum (resp., min-max) surface area (or $(d-1)$-volume) for a 2-box cover of $S \subset \Re^d$ can be $(1+\varepsilon)$-approximated in time $O(n+(1/\varepsilon)^d)$ (resp., $O(n+(1/\varepsilon)^{d-1})$).*

While the simple grid-based solution may give very good results in many cases, our goal is to give provably good results in *all* cases, including volume optimization in the very "thin" cases.

4. THE APPROXIMATION ALGORITHM FOR OPTIMIZING VOLUME

We focus our exposition around the 3-dimensional case of covering a set S of n points by a pair of axis-aligned boxes in order to minimize a combined measure (e.g., min-sum, min-max, min-union, etc) of their volume.

Let OPT denote, as before, an optimum pair of boxes, B_1 and B_2, that contain the entire point set S. There are exactly five geometric configurations (up to symmetric subcases) of pairs of boxes:

1. The two boxes are separable in one or more of the x, y, and z directions; see Figure 2(a).
2. One box completely pierces the other, so that no box contains any vertex of the other box; see Figure 2(b).
3. Exactly one edge of one box is fully contained within the other box; see Figure 2(c).[1]
4. Exactly one vertex of each box lies inside the other box; see Figure 2(d).
5. The two boxes have an "edge interaction," in which two edges of one box pass through the other, and vice versa; see Figure 2(e).

[1]Another way to specify this case is that exactly two vertices of one box lie inside the other box. These two vertices must be connected by an edge, otherwise they would not be the only vertices contained in the other box.

These cases cover all possible configurations of pairs of axis-aligned boxes. Indeed, partitioning cases according to how many vertices of one box are inside the other box, we get: 0 vertices (cases (a), (b), or (e)), 1 vertex (case (d)), 2 vertices (case (c)), 3, 5, 6, or 7 vertices (impossible), 4 or 8 vertices (possible for two boxes, but then the configuration would not be optimal).

We do not know in advance the structure of OPT; thus, we run separate procedures that search for the best coverage for different optimal configurations, each executed some constant number of times, in order to account for subcases that are the same, up to symmetry (e.g., reflection or relabeling of the coordinate axes).

Our algorithm begins by sorting (in $O(n \log n)$ time) all of the points along each of the three coordinates. We will discuss later a means of avoiding the sorting step.

4.1 The Separable Case

We first check for the best 2-box cover under the assumption that B_1 and B_2 are separable by a plane orthogonal to one of the coordinate axes. In this case, it is easy to find an optimal partition of S by a space-sweep algorithm in $O(n \log n)$ time. Assume, without loss of generality, that we sweep with an (x, y)-parallel plane, according to z-coordinate from bottom to top, in search of an optimal pair of boxes separated by a plane orthogonal to the z-axis. Initially, B_1 is empty and B_2 is the bounding box (computed in $O(n)$ time) containing all of the points of S (which are all initially above the sweep plane). When the sweep plane encounters a point $p_i \in S$, the point p_i switches from being in B_2 to being in B_1. Updating the two boxes can be done in $O(\log n)$ time per event if we store the points of each box in an appropriate data structure (e.g., a balanced binary tree for each of the three coordinate axes). During the sweep we keep track of the best combined measure of a pair of boxes during the sweep. Overall, this procedure requires $O(n \log n)$ time. More generally, in d dimensions, the time bound is $O(d^2 n \log n)$, where one factor d reflects the time needed to handle a single point, and the other factor d arises from repeating the procedure in each of the d directions.

4.2 The Nonseparable Cases

We state the following lemma for the more general d-dimensional setting. We say that box B is *large* in the x_i-direction if $x_i(B) \geq x_i(\mathrm{BB}(S))/2 = 1/2$.

LEMMA 6. *In the nonseparable cases, one of the two boxes of an optimal solution must have at least $\lceil d/2 \rceil$ of its d extents be large.*

PROOF. Since the two boxes B_1 and B_2 of an optimal solution are assumed not to be separable in any coordinate direction, the sums of the extents in each of the d directions is at least 1 (the corresponding full extent of $\mathrm{BB}(S)$): $x_i(B_1) + x_i(B_2) \geq x_i(\mathrm{BB}(S)) = 1$, for $i = 1, 2, \ldots, d$. Thus, one of the two boxes has at least $\lceil d/2 \rceil$ of its extents at least $1/2$. $\square$

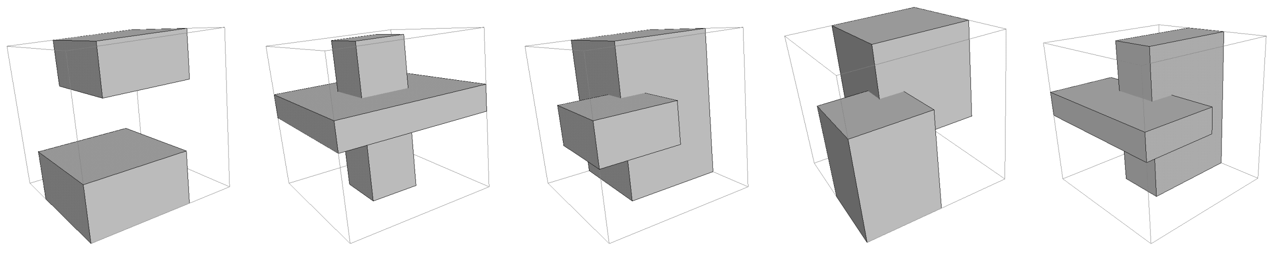
(a) Separable (b) Piercing (c) Edge in (d) Vertex in (e) Crossing

Figure 2: The five possible configurations of two covering boxes in $\Re^3$.

Specializing to the 3-dimensional case, we assume, without loss of generality, that B_1 is large in its x- and y-extents, and that the (x, y)-projection of B_1 is not contained fully in the (x, y)-projection of B_2. Note that B_2 may also be large in x- and/or y-extent, but it may be arbitrarily *small* in either or both of these extents.

4.2.1 Discretizing the (x, y)-projections

We proceed now with an analysis of subcases according to the (x, y)-projections of B_1 and B_2. We speak of "left" (min-x), "right" (max-x), "top" (max-y), and "bottom" (min-y) in the (x, y) projections of B_1, B_2, and BB(S).

We can assume that the left side of B_1 is flush with BB(S) (i.e., it is at $x = 0$, given our assumption about BB(S)). Let p_1 be a point of S with x-coordinate 0. Then, B_1 contains p_1, but B_2 does not, since there is no reason for them both to contain p_1, and B_2 is potentially smaller in measure if we delete p_1 from it.

In order to obtain a $(1 + \varepsilon)$-approximation, our general strategy will be to round the x- and y-coordinates that define B_1 *outwards* to a discrete set of choices in order that there are only a small (constant) number of possibilities to check. If we do this carefully, we will see that either $O(1/\varepsilon)$ or $O(1/\varepsilon^2)$ choices of "rounded out" subproblems suffice. Then, in the first case ($O(1/\varepsilon)$ choices), we can determine the z-extent of B_1 by examining either $O(n/\varepsilon)$ (in the min-sum case) or $O((logn)/\varepsilon)$ (in the min-max case) possibilities in the z-discretization.

$\bar{B}_1$ denotes the (x, y, z)-rounded out version of B_1; B_2 is the computed bounding box of all points not covered by $\bar{B}_1$.

In more detail, we do the following. Consider a decomposition of the (x, y)-projection of BB(S) into a k-by-k regular (rectangular) grid, where $k = \lceil 1/\varepsilon \rceil$. $\bar{B}_1$ is a box whose x- and y-sides lie along grid lines, with the left side containing the point $p_1 \in S$ and flush with the left side of BB(S).

Since B_1 has x-extent at least half of $x(\text{BB}(S))$ (it is "large"), we know that the x-extent of $\bar{B}_1$ is not much larger than that of B_1: $x(\bar{B}_1) \leq (1+\frac{2}{\varepsilon})\cdot x(B_1)$. A similar statement holds for the y-extent of $\bar{B}_1$.

We obtain the following subcases, according to how many of the four sides of BB(S) are flush with B_1 (see Figure 3):

(i) B_1 is flush with 4 sides of BB(S) in the (x, y)-projection. In this case, B_1 is already maximal in its (x, y)-projection, so there is no rounding out needed to determine the (x, y)-projection of $\bar{B}_1$ (it is the same as the (x, y)-projection of B_1 and of BB(S)). This corresponds to the piercing case (Figure 2(b)). There remain two z-values to determine (the z-extent of $\bar{B}_1$), which is done in time $O(n/\varepsilon)$ as described below.

(ii) B_1 is flush with exactly 3 sides of BB(S) in the (x, y)-projection. In this case, B_1 is already maximal in its y-projection, so we only have to round outwards in x to determine the right side of $\bar{B}_1$: there are $O(1/\varepsilon)$ choices. This corresponds to case (c) or case (e) of Figure 2(b). In case (c) there is one z-value remaining to determine, which is easily done in $O(n)$ time; in case (e) there are two z-values to determine the z-extent of $\bar{B}_1$, which is done in time $O(n/\varepsilon)$ as described below.

(iii) B_1 is flush with exactly 2 sides of BB(S) in the (x, y)-projection. In this case, we have to round outwards in two directions, according to which two sides of B_1 are not flush with the corresponding sides of BB(S). There are $O(1/\varepsilon^2)$ choices.

In the top subcase of Figure 3 (adjacent flush sides), the situation corresponds to case (c) or case (d) of Figure 2; in the bottom subcase of Figure 3 (opposite flush sides), the situation is case (b) of Figure 2. In case (d), there is only one z-value of $\bar{B}_1$ to determine, since the other is flush; this is done in time $O(n)$. In case (b), B_1 is either pass-through in x and in z (it is the "plate" being pierced in case (b)), in which case there is no z-value of $\bar{B}_1$ to determine, or B_1 is pass-through only in x (it is the one piercing the plate), in which case B_2 is pass-through in y and in z, and we can reverse the roles of B_1 and B_2 (B_2 is large in y and z, so this case is considered separately in the other projections).

In case (c), there is more care needed. If B_1 is pass-through in z, we are done (rounding out in (x, y) suffices to determine $\bar{B}_1$). Thus, assume that B_2 is the one that is pass-through in z. If the rounding out of B_1 in the k-by-k grid causes it to be pass-through in x, then there are only $O(1/\varepsilon)$ choices for the other y-coordinate of $\bar{B}_1$, leaving $O(n/\varepsilon)$ time to determine the z-extent by the methods below. On the other hand, if rounding out B_1 in x does not cause it to be pass-through in x, then we know that B_2 has "substantial" x-extent (at least $1/\varepsilon$), so we switch over to discretizing the (x, y)-projection of B_2, using $O(1/\varepsilon^2)$ choices for its x-extent and n (or $\log n$, in the min-max case) choices for its y-extent, in much the same way that we determine z-extents below.

(iv) B_1 is flush with exactly 1 side of BB(S) in the (x, y)-projection. The situation corresponds either to case (c) or case (e) of Figure 2. If it is case (e), then B_1 is pass-through in z, so we can consider $O(1/\varepsilon^3)$ (or, in the min-max case, $O((\log n)/\varepsilon^2)$) choices for the y-extent and x-extent. If it is case (c), then one facet of B_1 is flush in z, and it appears that in (x, y) we must round outwards in three directions: $+x$, $+y$, and $-y$. This would lead to $O(1/\varepsilon^3)$ choices in (x, y), after which we may need n choices (or $\log n$, in the min-max case) in z. Our goal is to do better.

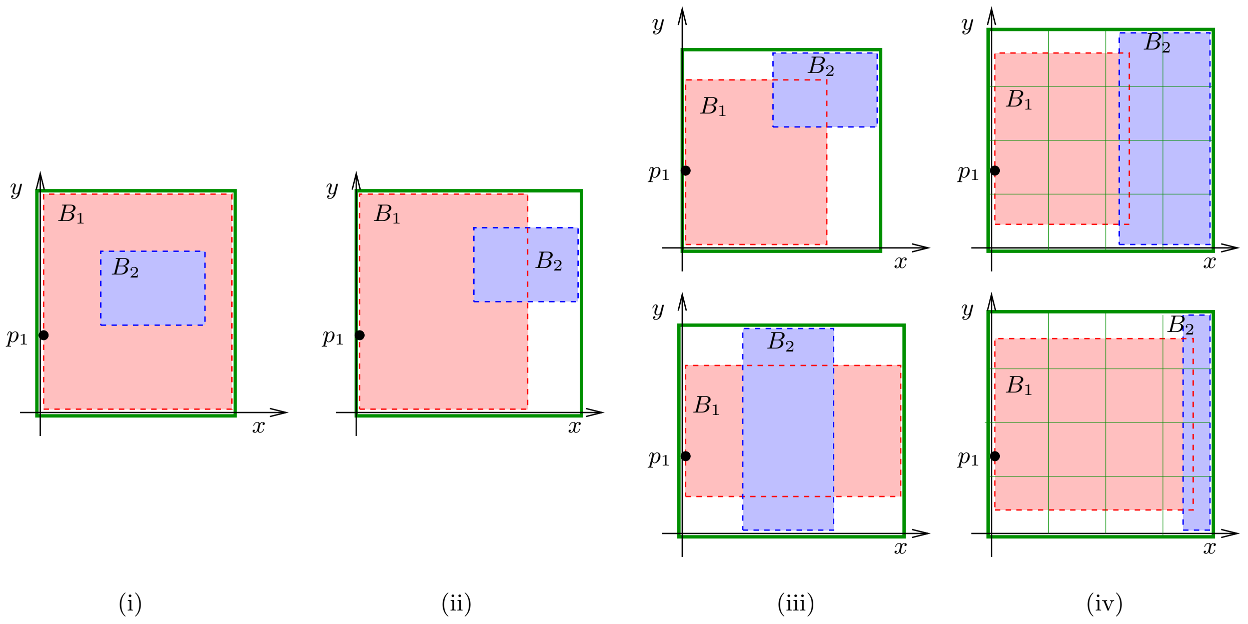

Figure 3: Cases (i)–(iv) for the (x, y)-projections of B_1 and B_2. The box $\mathrm{BB}(S)$ is drawn with a thick border.

We reason as follows. If the right side of B_1 lies within $1/k$ of the right side of $\mathrm{BB}(S)$ (i.e., the right side is to the right of the next-to-last vertical grid line, as in the lower subcase of Figure 3), then the rounding out in x takes us to case (iii), and there are only $O(1/\varepsilon^2)$ choices of top/bottom in the rounding out in y. On the other hand, if the right side of B_1 lies *further than* $1/k$ from the right side of $\mathrm{BB}(S)$ (as in the above subcase of Figure 3), then we know that B_2 must have "substantial" (more than $1/k$) x-extent, since B_1 and B_2 are assumed not to be separable in x. Furthermore, B_2 is pass-through in y. Thus, in this case, we switch the roles of B_1 and B_2 and consider discretizations of the (x, y)-projection of B_2: we consider $O(1/\varepsilon^2)$ choices, i/k^2, for $i = 1, 2, \ldots, k^2 - k\}$, for the left side of a rounded-out box, $\bar{B}_2$. For each of these choices of (x, y)-projection of $\bar{B}_2$, we compute (in time $O(n/\varepsilon)$) an approximately optimal choice of z-extent of $\bar{B}_2$, using the procedure described below. (Then, B_1 will be taken to be the bounding box of those points not covered by the rounded out $\bar{B}_2$.)

4.2.2 Determining the z-extent of $\bar{B}_1$

After having argued that the choices of (x, y)-projection of $\bar{B}_1$ can be discretized, we turn our attention to determining the z-values, $\bar{z}_{min}$ and $\bar{z}_{max}$, that define our outer approximation $\bar{B}_1$. Of course, we could consider all possible pairs of points that may determine $\bar{z}_{min}$ and $\bar{z}_{max}$, but our goal is to avoid quadratic complexity in n. It is also important to notice that we cannot assume that the z-extent of $\bar{B}_1$ is "large" in any sense; it may be arbitrarily close to 0, and we cannot afford to round it out, say, to an interval determined by two z-values of the form i/k, $i = 0, 1, \ldots, k$, as this could increase the z-extent (and therefore the volume) of B_1 by a factor that is arbitrarily large.

In order to determine the z-extent, $[\bar{z}_{min}, \bar{z}_{max}]$, of $\bar{B}_1$ we make use of the knowledge of the point $p_1 = (x_1, y_1, z_1) \in S$ that is known to be contained in B_1. In particular, we know that z_1 is contained in the z-extent, $[z_{min}, z_{max}]$, of B_1. We assume that $z_{max} - z_1 \geq z_1 - z_{min}$; separately and analogously we can handle the case in which $z_1 - z_{min} > z_{max} - z_1$.

We consider each possible choice of $\bar{z}_{max}$, iterating over the z-coordinates of S, starting from z_1 and increasing. (Recall that we pre-sorted S by each of its coordinates.) For each choice of $\bar{z}_{max}$, we consider up to $k = \lceil 1/\varepsilon \rceil$ choices of $\bar{z}_{min}$: $\{z_1 - i \cdot \frac{\bar{z}_{max} - z_1}{k} : i = 1, 2, \ldots, k\}$. Let $z^{(1)}, z^{(2)}, \ldots, z^{(k)}$ denote the choices for $\bar{z}_{min}$. Note that as we advance $\bar{z}_{max}$ over larger and larger values of z, each of the $z^{(j)}$ choices decreases monotonically.

We need to argue that we can advance $\bar{z}_{max}$ (and, correspondingly, the $z^{(j)}$'s) efficiently, and update each of the two bounding boxes efficiently.

We do this by precomputing (in time $O(n \log n)$) the rectilinear hull of the projection of S into each of the three planes ((x, y), (x, z), and (y, z)). We maintain pointers associated with $\bar{z}_{max}$ and each one of the k choices of $\bar{z}_{min}$ corresponding to $z^{(j)}$. As these pointers advance, we can determine the new x- and y-extents of the bounding box $B_2^{(a)}$ of those points of S whose (x, y)-projection lies within the (x, y)-projection of $\bar{B}_1$ and whose z-coordinate is greater than $\bar{z}_{max}$; similarly, we determine the bounding box $B_2^{(b)}$ of those points of S whose (x, y)-projection lies within the (x, y)-projection of $\bar{B}_1$ and whose z-coordinate is less than $z^{(j)}$, for each $j = 1, 2, \ldots, k$. Refer to Figure 4. The bounding box B_2, for each choice of $[z^{(j)}, \bar{z}_{max}]$, is obtained as the bounding box of the three bounding boxes $B_2^{(a)}$, $B_2^{(b)}$, and $B_2^{(c)}$, where $B_2^{(c)}$ is the bounding box of those points of S whose (x, y)-projections lie outside the (x, y)-projection of $\bar{B}_1$. (This box $B_2^{(c)}$ is computed once for each of the choices of (x, y)-projection of $\bar{B}_1$.)

The total time required is $O(n)$ (after the initial sorting of S by x-, y-, and z-coordinates). As we discover the coordinates of the $O(n)$ boxes $\bar{B}_1$, $\bar{B}_2^{(a)}$, and $\bar{B}_2^{(b)}$, for each choice of $\bar{z}_{max}$ and for each of the corresponding $O(1/\varepsilon)$ choices of

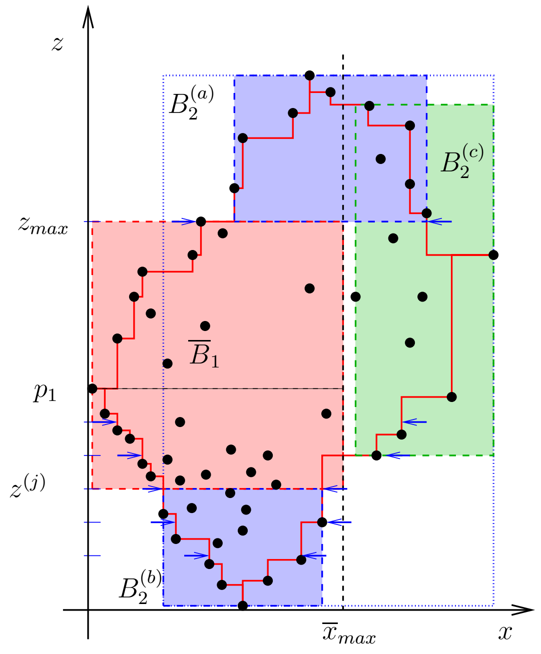

Figure 4: Illustration of the (x, z)-projection used in determining the bounding boxes as the z-extent is varied. A similar figure is drawn in the (y, z)-plane.

the $z^{(j)}$'s, we keep track of the min-max or min-sum of the measures of the pairs of boxes, $(\bar{B}_1, B_2)$, enumerated.

We can improve the algorithm for the min-max case. Instead of enumerating all choices of $\bar{z}_{max}$, we use the following monotonicity structure to perform a binary search:

LEMMA 7. *For a fixed choice of rounded out (x, y)-projection, and a fixed choice of $j \in \{1, 2, \ldots, k\}$, the measure of $\bar{B}_1$ is monotonically increasing in $\bar{z}_{max}$, and the measure of $\bar{B}_2^{(a)}$ and of $\bar{B}_2^{(b)}$ is monotonically decreasing in $\bar{z}_{max}$.*

We can apply Lemma 7 as follows. For a given choice of (x, y)-projection ($O(1/\varepsilon^2)$ choices), we do $O(1/\varepsilon)$ binary searches (one for each choice of j) in order to determine the value of $\bar{z}_{max}$ to minimize the maximum measure of the two boxes $\bar{B}_1$ and B_2. We can test a value, z', of $\bar{z}_{max}$ in time $O(1)$, as follows. We compute the measures of the corresponding $\bar{B}_1$, $\bar{B}_2^{(a)}$, $\bar{B}_2^{(b)}$, and B_2, and compare the measure, μ_1, of $\bar{B}_1$ to the measure, μ_2, of B_2. (The measures are found in $O(1)$ time if, when we build the rectilinear hulls of S in each of the three projections, we keep track of the projection onto the hull boundary of each point of S interior to the hull.) If $\mu_1 > \mu_2$, we restrict the range of $\bar{z}_{max}$ to be at most z'; otherwise, we restrict the range of $\bar{z}_{max}$ to be at least z'.

Avoiding the $O(n \log n)$ Sort. Instead of doing "exact" sorting of the points in $O(n \log n)$ time, it suffices to do "approximate" sorting. In particular, instead of sorting the x-coordinates, we can determine, in $O(n)$ time, the point, $p^{(i)}$, of S with smallest x-coordinate in the range $[i/k, 1]$. Doing this for each $i = 1, 2, \ldots, k$ takes overall time $O(kn) = O(n/\varepsilon)$ to produce the list $p^{(1)}, p^{(2)}, \ldots, p^{(k)}$ of x-sorted representatives of S.

In summary, we state:

THEOREM 8. *The min-max volume 2-box cover problem has a $(1 + \varepsilon)$-approximation algorithm that runs in $O(n \min\{\log n, 1/\varepsilon\} + (\log n)/\varepsilon^2)$ time, for any fixed $\varepsilon > 0$. The min-sum version can be solved in $O(n \min\{\log n, 1/\varepsilon\} + n/\varepsilon^2)$ time.*

PROOF. (Sketch) Let B_1 and B_2 be an optimal pair of boxes. If they are separable, then we discover this pair during the sweep described above. If they are not separable, then one of the two boxes must have large extent in two coordinates (by Lemma 6); assume this box is B_1. Then, in the search we just described, we consider all possible roundings of B_1 outward by at most $1/\varepsilon$ in each coordinate that is not already flush against $\text{BB}(S) = U$; in some of the cases, we switched the role of B_1 and B_2 and search instead on an appropriate discretization of the extents of B_2. In all cases, the outward rounding is done so that no extent goes up by more than a factor $(1 + \varepsilon)$. Thus, the volume of the box under consideration (B_1, or sometimes B_2) goes up by at most a factor $(1 + \varepsilon)^3$, which is arbitrarily close to 1. $\square$

Higher Dimensions. In d dimensions, our results are summarized in the following theorem.

THEOREM 9. *In dimension $d \geq 2$, the min-max volume 2-box cover problem has a $(1 + \varepsilon)$-approximation algorithm that runs in time*

$$O(d^2 n \min\{\log n, 1/\varepsilon\} + \frac{1}{\varepsilon^{\lceil d/2 \rceil}} \binom{d}{\lceil d/2 \rceil} d^2 n^{\lfloor d/2 \rfloor - 1} \log n),$$

for any fixed $\varepsilon > 0$. The min-sum version can be solved in time

$$O(d^2 n \min\{\log n, 1/\varepsilon\} + \frac{1}{\varepsilon^{\lceil d/2 \rceil}} \binom{d}{\lceil d/2 \rceil} d^2 n^{\lfloor d/2 \rfloor}).$$

PROOF. (Sketch) The first term in both time complexities is due to the d sortings (or approximate sortings) of the points, where each point comparison takes $O(d)$ time. The second term represents the same process as in three dimensions, only now we choose $\lceil d/2 \rceil$ out of the d directions, since, by Lemma 6, one of the two boxes must have at least $\lceil d/2 \rceil$ of its extents be large. $\square$

5. EXTENSIONS AND OTHER RESULTS

5.1 Min-Union 2-Box Cover

In some applications we are interested in minimizing the size (e.g., volume) of the union $B_1 \cup B_2$. All of our results for the min-sum case carry over to min-union case, since we in fact enumerate all cases that may correspond to a slight rounding out of either or both of the boxes. In particular, we can compute the exact solution in time $O(n^3)$ (in $\Re^3$). Also, since the sum of the volume errors is small (at most ε times OPT) in our approximation algorithm, we know that the volume of the union is also approximated with error at most ε times OPT. The speed-up we get for the min-max case does not immediately apply to the min-union case, though. However, the min-max results do apply to give a 2-approximation for the min-union (volume), since

$$vol(B_1 \cup B_2) \leq vol(B_1) + vol(B_2) \leq 2 \cdot \text{min-max}(S).$$

THEOREM 10. *The min-union (volume) 2-box cover problem in $\Re^3$ has a $(1 + \varepsilon)$-approximation algorithm that* *runs in time $O(n \min\{\log n, 1/\varepsilon\} + n/\varepsilon^2)$ time, for any fixed $\varepsilon > 0$. It has a 2-approximation that runs in time $O(n \min\{\log n, 1/\varepsilon\} + (\log n)/\varepsilon^2)$.*

5.2 2-Box Cover of Polyhedra

We consider now the case in which S is given as a set of polygons (e.g., the facets of a polyhedral solid model), or more generally as a set of sets of points (e.g., the vertices of a polygon form one such set). Our goal is to partition S into two subsets S_1 and S_2, deciding for each polygon of S which subset should contain it, in order to minimize a combined measure of the two bounding boxes, B_1 and B_2. This version of the 2-box cover models most closely the main step in a top-down construction algorithm for a bounding volume hierarchy using AABBs.

First, note that we can replace each polygon or element of S by its bounding box; thus, it suffices to handle the case of S being a set of (possibly overlapping) axis-aligned boxes $\{b_1, \ldots, b_n\}$.

Consider one of the choices of (x, y)-projections for $\bar{B}_1$ that is enumerated during our main algorithm; all boxes whose projection lies fully inside the rectangle are candidates to be contained (the others must go to B_2). Now, we must determine the z-extent. There is now a "core" box, say b_1, instead of the core point p_1. In the z-discretization step, we can sweep upwards from the top of b_1 and downwards from the bottom of b_1. For each candidate choice of z_{max} (which is the "top" of some box b_i) we consider the possible $z^{(j)}$ values. There may be some boxes b_j that partially overlap or fully pass through $\bar{B}_1$; these are forced to be in B_2, of course. Thus, we need to enhance the data structure based on the rectilinear hulls (computed in each of the three projections): Consider the (x, z) projection. We construct the rectilinear hull of the projected boxes. But we need to be able to do the following: For a candidate z_{max}, at the top edge of some b_i, we need to determine the bounding box of all of those boxes b_j that extend *across* the line $z = z_{max}$. In order to do this, we enhance our data structure: During the sweep to construct the rectilinear hull, the "status" of the sweep keeps track of the bounding box of the rectangles the line stabs. (These will have to be assigned to B_2.) Record the status at each of the n events. Then, we can use this preprocessed data to advance z_{max} in $O(1)$ time per step, and correspondingly advance the $z^{(j)}$'s downwards as well. The details appear in the full paper.

THEOREM 11. *The min-max (volume) 2-box cover problem for covering a set of polygons or polyhedra in $\Re^3$ has a $(1 + \varepsilon)$-approximation algorithm that runs in $O(n \min\{\log n, 1/\varepsilon\} + (\log n)/\varepsilon^2)$ time, for any fixed $\varepsilon > 0$. The min-sum and min-union (volume) versions can be solved in $O(n \min\{\log n, 1/\varepsilon\} + n/\varepsilon^2)$ time. The min-sum (resp., min-max) surface area approximation runs in time $O(n + (1/\varepsilon)^3)$ (resp., $O(n + (1/\varepsilon)^2)$).*

5.3 Cardinality Constraints: Balancing the Partition

We consider now the variant of the 2-box cover in which we want to partition S into S_1 and S_2, each having at least αn elements, for a specified constant $0 < \alpha \leq 0.5$. We call this the *α-constrained 2-box cover* problem. (A related question is to constrain each of B_1 and B_2 to contain at least αn elements of S; in this case, we can allow any α between 0 and 1.) We limit our discussion here to the min-max/min-sum coverage of points in $\Re^3$ to optimize surface area of the bounding boxes. The simple grid-based method can be applied, as before, but the time bound goes up by $1/\varepsilon^2$ in order to account for the fact that once B_1 is specified approximately, we cannot simply take B_2 to be the bounding box of the remainder, since this may not contain enough points of S to satisfy the constraint that $|S_2| \geq \alpha n$. Thus, we do an additional search, at cost of $1/\varepsilon^2$ per candidate B_1. In the full paper we prove:

THEOREM 12. *The min-sum (resp., min-max) surface area for an α-constrained 2-box cover of $S \subset \Re^3$ can be $(1 + \varepsilon)$-approximated in time $O(n + (1/\varepsilon)^5)$ (resp., $O(n + (1/\varepsilon)^4)$).*

Remark: We have been unable so far to find a near-linear time approximation scheme for the α-constrained 2-box cover using the volume measure. (Our current bound is near-cubic.)

5.4 Building Bounding Volume Hierarchies

We can apply our methods to yield bounding volume hierarchies for a set S of points or polyhedra. Here, we give only a brief sketch.

Starting with the full set at the root node, we partition S into S_1 and S_2 using our approximation algorithm. Continuing down the tree, we apply the algorithm at every node of the hierarchy. Our nearly-linear time bound applies at each level of the tree, yielding an overall time bound that is at most the height of the tree times the complexity of the 2-box cover algorithm. By using the balanced partition methods described above (for min-max or min-sum surface area), we are able to bound the height of the tree by $\log n$, where the base of the logarithm depends on the choice of α. This gives

THEOREM 13. *For a given set S of points or polyhedra in $\Re^3$, one can construct a bounding volume hierarchy, of height $O(\log n)$, with each partition of the set at each node being done with an approximation bound of $(1+\varepsilon)$ times optimal (in min-max or min-sum, surface area). The running time is nearly linear in n for any fixed ε.*

6. CONCLUSION

In this paper we provide several new algorithms for the problem of covering a set of points or polyhedra by a pair of axis-aligned boxes in order to minimize the sum or the maximum of the measures of the two boxes, under a variety of different possible measures. Our algorithms are simple enough that they should be amenable to implementation.

Possible further research directions include the following:
(1) Can we extend our results to yield approximation results for covering by two oriented bounding boxes? More generally, can we cover by a pair of k-dops or minimize the measures of the *convex hulls* of the two sets S_1 and S_2?
(2) Can we extend our results to covering with more than two boxes?
(3) Can we apply core-set methods to obtain better dependence on d in higher dimensions?
(4) Are the algorithms practical, and do they lead to improvements in the performance of bounding volume hierarchies?

Acknowledgments

The authors thank David Mount and Micha Sharir for earlier discussions on the problems studied here.
This research was partially supported by grants from the National Science Foundation (CCR-0098172, ACI-0328930, CCF-0431030, CCF-0528209), the U.S.-Israel Binational Science Foundation (2000160), Metron Aviation, NASA Ames (NAG2-1620), and the European FP6 Network of Excellence AIM@SHAPE (506766).

7. REFERENCES

[1] P. K. Agarwal, M. de Berg, J. Gudmundsson, M. Hammar, and H. J. Haverkort. Box-trees and r-trees with near-optimal query time. In *Proc. 17th Ann. ACM Symp. on Comp. Geometry*, Medford, MA, 124–133, June 2001.

[2] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surv.*, 30:412–458, 1998.

[3] H. Alt, E. M. Arkin, H. Brönnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, and K. Whittlesey. Minimum-cost coverage of point sets by disks. In *Proc. 22th Ann. ACM Symp. on Computational Geometry*, Sedona, AZ, June 2006, to appear.

[4] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. *Proc. 30th Ann. ACM Symp. on Theory of Computing*, 106–113, 1998.

[5] J. Arvo and D. Kirk. Fast ray tracing by ray classification. In *Proc. SIGGRAPH*, 55–63, 1987.

[6] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, and A. Tal. BOXTREE: A hierarchical representation for surfaces in 3D. *Comput. Graph. Forum*, 15(3):C387–C396, C484, Sep. 1996. Proc. Eurographics '96.

[7] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD Conf. on Management of Data*, 322–331, 1990.

[8] S. Bespamyatnikh and M. Segal. Covering a set of points by two axis-parallel boxes. *Inf. Process. Lett.*, 75(3):95–100, 2000.

[9] V. Bilò, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *Proc. 13th European Symp. on Algorithms*, 2005.

[10] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proc. 34th Ann. ACM Symp. on Theory of Computing*, 250–257, 2002.

[11] M. Charikar and R. Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2):417–441, 2004.

[12] Z. Drezner. On the rectangular p-center problem. *Naval Res. Logist. Q.*, 34:229–234, 1987.

[13] J. Goldsmith and J. Salmon. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications*, 7:14–20, 1987.

[14] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-tree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30:171–180, 1996.

[15] The GNU triangulated surface library (GTS), `http://gts.sourceforge.net`.

[16] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conf. on Principles Database Systems*, 47–57, 1984.

[17] T. He. Fast collision detection using QuOSPO trees (color plate S. 222). In *Proc. ACM Symp. on Interactive 3D Graphics*, 55–62, 1999.

[18] M. Held, J. Klosowski, and J. S. B. Mitchell. Real-time collision detection for motion simulation within complex environments. In *Proc. ACM SIGGRAPH Visual Proceedings*, page 151, 1996.

[19] M. Held, J. T. Klosowski, and J. S. B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Proc. 7th Canadian Conf. on Computational Geometry*, 205–210, 1995.

[20] J. Hershberger. Minimizing the sum of diameters efficiently. *Computational Geometry: Theory and Applications*, 2(2):111–118, 1992.

[21] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Trans. on Visualizat. Comput. Graph.*, 1(3):218–230, Sep. 1995.

[22] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. on Graphics*, 15(3):179–210, 1996.

[23] P. Jimnez, F. Thomas, and C. Torras. 3D Collision detection: A survey. *Computers and Graphics*, 25(2):269–285, 2001.

[24] T. L. Kay and J. Kajiya. Ray tracing complex scenes. *Computer Graphics*, 20(4):269–278, 1986.

[25] J. Klosowski, M. Held, J. S. B. Mitchell, K. Zikan, and H. Sowizral. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. on Visualizat. Comput. Graph.*, 4(1):21–36, 1998.

[26] J. T. Klosowski. *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments.* Ph.D. thesis, Dept. of Applied Mathematics and Statistics, Stony Brook University, 1998.

[27] P. Kumar, J. S. B. Mitchell, and A. Yıldırım. Approximate minimum enclosing balls in high dimensions using core-sets. *The ACM J. of Experimental Algorithmics*, 8, 2003.

[28] N. Lev-Tov and D. Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47:489–501, 2005.

[29] M. Lin and S. Gottschalk. Collision detection between geometric models: A survey. In *Proc. of IMA Conf. on Mathematics of Surfaces*, 1998.

[30] I. J. Palmer and R. L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics Forum*, 14(2):105–116, June 1995.

[31] J. Ponce and O. Faugeras. An object centered hierarchical representation for 3D objects: The prism tree. *Computer Vision, Graphics, and Image Processing*, 38:1–28, 1987.

[32] H. Samet. *Spatial Data Structures: Quadtrees, Octrees, and Other Hierarchical Methods.* Addison-Wesley, Reading, MA, 1989.

[33] M. Sharir and E. Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th Ann. ACM Symp. on Computational Geometry*, 122–132, May 1996.

[34] SOLID home page, `http://www.win.tue.nl/cs/tt/gino/solid/`.

[35] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *J. Graphics Tools*, 2(4):1–13, 1997.

[36] G. van den Bergen. A fast robust GJK implementation for collision detection of convex objects. *J. Graphics Tools*, 4(2):7–25, 1999.

[37] H. Weghorst, G. Hooper, and D. P. Greenberg. Improved computational methods for ray tracing. *ACM Trans. on Graphics*, 3(1):52–69, 1984.

[38] A. Wilson, E. Larsen, D. Manocha, and M. C. Lin. Partitioning and handling massive models for interactive collision detection. *Computer Graphics Forum (Eurographics '99)*, 18(3), 319–330, 1999.

[39] G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proc. IEEE Virtual Reality Ann. Int. Symp.*, Atlanta, GA, 90–97, March 1998.

[40] G. Zachmann. Minimal hierarchical collision detection. In *ACM Virtual Reality Software and Technology*, 121–128, Nov. 2002.