An Efficient Divide-and-Conquer Approximation Algorithm for Hyperrectangular Partitions‡

Teofilo Gonzalez, Mohammadreza Razzazi,
Department of Computer Science
The University of California
Santa Barbara, CA 93106

and

Si-Qing Zheng
Computer Science Department
Louisiana State University
Baton Rouge, LA 70803-4020

EXTENDED ABSTRACT

The problem of partitioning a polygon is fundamental in Computational Geometry, and as a result of this it has been studied under many different objective functions. Lingas *et. al.* ([LPRS]) investigated the problem of partitioning the inside of a rectilinear polygon with holes into rectangles, the goal was to obtain a partition with minimum length of partitioning edges. In VLSI design, the problem of dividing routing regions into channels can be reduced to this partitioning problem ([R]).

A rectilinear *boundary* is a simple polygon with the additional constraint that all of its sides are either parallel or perpendicular to each other. A *hole* is a simple rectilinear polygon located inside the rectilinear boundary. There cannot be holes inside a hole. A single point inside the boundary is called a *degenerate hole*. A *figure* is a rectilinear boundary which may contain an arbitrary number of nonoverlapping holes. A *rectangular partition* of a figure is a set of line segments lying within its boundary and not crossing any non-degenerate hole so that when added to the figure, the area not enclosed by holes is partitioned into rectangles that do not contain as interior points degenerate holes. The partitioning line segments are called *edges*. For every problem instance I and every set of edges $E(I)$ in a feasible solution, we use the function $L(E(I))$ to represent the sum of the length of the edges in $E(I)$. A minimum edge length partition of a figure is a rectangular partition with least $L(\cdot)$ among all rectangular partitions. An approximation algorithm is said to have an *approximation bound* of C if for every problem instance, I, $L(E) \leq C\, L(O)$, where E is the set of edges in the solution generated for I by the approximation algorithm, and O is the set of edges in an optimal solution for I.

Lingas *et. al.* [LPRS] showed that when there are interior holes, the problem of finding minimum edge length rectangular partitions is NP-hard. The problem remains NP-hard even when all the holes are degenerate holes and the boundary is a rectangle [LPRS]. Hereafter we shall refer to this restricted problems as the $RG-P_2$ *problem.* Several approximation algorithms for the general problem exist (see [L], [DC], [L1] and [L2]). The algorithms with the smallest approximation bound are the ones reported in [L1] and [L2]. Furthermore, the algorithm given in [L2] uses the algorithm reported in [GZ1] for the $RG-P_2$ problem as a sub-procedure. Other approximation algorithms for the $RG-P_2$ problem appear in [GZ1], [GZ2], [GZ3], [DPS] and [GRSZ]. In table I we summarize the currently best approximation algorithms for the $RG-P_2$ problem.

‡ This research was supported in part by the National Science Foundation under Grant DCR - 8503163

Approximation algorithms for partitioning a rectangle			
Approximation Bound	Time Complexity Bound	Reference	Method
$3+\sqrt{3}$	$O(n \log n)$	[GZ1], [L2]	divide and conquer
3	$O(n^4)$	[GZ2]	transformation
1.75	$O(n^5)$	[GZ4]	dynamic programming

Let P be a set of points located inside a hyperrectangle (rectangle if $d = 2$) R. The $RG-P_d$ problem, which is a generalization of the $RG-P_2$ problem to d dimensions, consists of partitioning R into hyperrectangles (rectangles if $d = 2$) by introducing a set of hyperplane (line if $d = 2$) segments of least total (d-1)-volume (length if $d = 2$). Each hyperrectangle (rectangle if $d = 2$) in a valid partition cannot contain points from P as interior points. For every problem instance I and every set of edges $E(I)$ in a feasible solution, we use the function $V_{d-1}(E(I))$ to represent the sum of the (d-1)-volume of the hyperplane segments in $E(I)$. This problem has been recently been considered in [GRSZ] where a dynamic programming approximation algorithm based on guillotine partitions is analyzed. Their algorithm takes $O(n^{2d+1})$ time and generates solutions within $2d - 4 + 4/d$ times the optimal solution value. An application for the $RG-P_3$, is discussed in [GRSZ].

In this paper we present a divide-and-conquer algorithm that takes $O(dn \log n)$ time and it generates solutions similar to the ones generated by the algorithm in [GZ1]. The main difference between the result in the paper and the one in [GZ1] is that the new approximation bound is smaller (four instead of $3 + \sqrt{3}$), the new algorithm is simpler (we only introduce one cut at each step), the proof is much simpler than the previous one (there are fewer cases and the proof for each case is simpler), and the new algorithm has the approximation bound $2d$ for all $d \geq 2$. With respect to the results in [GRSZ], the algorithm in this paper is faster ($O(dn \log n)$ instead of $O(n^{2d+1})$), but the algorithm in [GRSZ] always generates solutions which are closer to optimal.

THE ALGORITHM

The $RG-P_d$ problem is formally defined by $I = (R = (o, X), P)$, where o and X define a hyperrectangle or boundary R ($o = (o_1, o_2, ..., o_d)$ is the "lower-left" corner of the boundary (*origin* of I), and $X = (X_1, X_2, ..., X_d)$ are the dimensions of the boundary) in d-dimensional Euclidean space (E^d), and $P = \{ p_1, p_2, ..., p_n \}$ is a set of points (degenerate holes) inside hyperrectangle R. We define $X_{\bar{i}}$ as $X_1 \cdot X_2 \cdot ... \cdot X_{i-1} \cdot X_{i+1} \cdot ... \cdot X_d$, and $X_{i,j}$ as $X_i \cdot X_{i+1} \cdot ... \cdot X_j$, for $i \leq j$. We shall refer to the d dimensions (or axes) of E^d by the integers 1,2, ..., d (in 2-space we have the first dimension (x-dimension, x-axis, or 1-axis) and the second dimension (y-dimension, y-axis, or 2-axis))

We use E_{apx} to denote the solution generated by our algorithm. Initially E_{apx} is empty. Our algorithm, which we refer to as procedure PARTITION, first checks if P is empty and if so, it returns. Otherwise, it introduces a mid-cut or an end-cut after relabeling the axes so the $X_1 \geq X_2 \geq ... \geq X_d$. A *mid-cut* is a hyperplane segment orthogonal to the 1-axis that intersects the center of the hyperrectangle and an *end-cut* is a hyperplane segment orthogonal to the 1-axis that contains either the "leftmost" or the "rightmost" point in P. A mid-cut is introduced when the two resulting subproblems have at least one point each. Otherwise, an end-cut is introduced. The end-cut intersects the leftmost point if such a point is not located to the left of the center of the hyperrectangle, otherwise the end-cut intersects the rightmost point. We shall refer to the two resulting subproblems as $I_1=(R_1,P_1)$ and $I_2=(R_2,P_2)$. The length along the first dimension of R_1 (R_2) is referred to by X_1' (X_1'').

We define the function, LB(I), by taking a "portion" of the (d-1)-volume at each step of our recursive algorithm as follows.

$$LB(I) = \begin{cases} 0 & P = \varnothing \\ X_{\bar{1}} & \text{an end-cut is introduced, } P_1 = \varnothing \text{ and } P_2 = \varnothing \\ \mathrm{LB}(I_1) + \mathrm{LB}(I_2) & \text{an mid-cut is introduced} \\ \mathrm{LB}(I_1) + \min\{ X_{\bar{1}}, X_1' X_{3,d} \} & \text{an end-cut is introduced, } P_1 \neq \varnothing \text{ and } P_2 = \varnothing \\ \mathrm{LB}(I_2) + \min\{ X_{\bar{1}}, X_1'' X_{3,d} \} & \text{an end-cut is introduced, } P_1 = \varnothing \text{ and } P_2 \neq \varnothing \end{cases}$$

Lemma 1: For any problem instance I, LB(I) $\leq V_{d-1}(E_{opt}(I))$.
Proof: For brevity the proof is not included.

□

Assume that $X_1 \geq X_2 \geq \ldots \geq X_d$. For convenience we define $X_0 = X_1$ and $X_{d+1} = X_d / 3$. Note that $X_1 \leq 2X_1$ and $X_1 > 2X_{d+1}$ for all I. A problem instance $I = (o , X , P)$ is said to be of *type i* ($0 \leq i < d$) if $X_1 \leq 2X_{i+1}$ and $X_1 > 2X_{i+2}$. We define the CARRY function as follows: $(d-i)X_{\overline{i+1}} + \sum_{j=1}^{i} X_{\bar{j}}$ if I is type i for $0 < i \leq d$-1, and $(2d-1) X_{\bar{1}}$ if I is type 0. One may visualize the algorithm as follows. Whenever a hyperplane segment is introduced by the algorithm (mid-cut or end-cut) it is colored red, and when a lower bound corresponding to a hyperplane segment is detected it is marked blue. Our approach is to bound the sum of the (d-1)-volume of all the red segments by $2d$ times the sum of the (d-1)-volume of the blue segments. The idea behind our proof is that the CARRY function corresponds to the (d-1)-volume of some red segments which have not yet been accounted for by blue segments and which will be accounted for by some blue segments which will appear in recursive calls from I. The worst case bound for our approximation algorithm is given by theorem 1.

Theorem 1: For any problem instance I,
(i) $V_{d-1}(E_{apx}(I)) + \mathrm{CARRY}(I) \leq 2d \cdot \mathrm{LB}(I)$, and
(ii) $V_{d-1}(E_{apx}(I)) + \mathrm{CARRY}(I) \leq 2d \cdot V_{d-1}(E_{opt}(I))$.
(iii) $V_{d-1}(E_{apx}(I)) \leq 2d\ V_{d-1}(E_{opt}(I))$.
Proof: For brevity the proof is omitted.

□

For all $\varepsilon > 0$, there are problem instances such that $V_{d-1}(E_{apx}) = 2d\ V_{d-1}(E_{opt}) - \varepsilon$. For brevity we do not discuss them in detail. A straight forward implementation of algorithm PARTITION requires $O(dn^2)$ time, where n is the number of points in P. However, procedure PARTITION can be implemented to take $O(dn \log n)$ time. The idea is to translate each point into a tuple of d integers in the range $[1,n]$ and to represent the set of points by a multilinked data structure which can be easily traversed in several directions.

Theorem 2: Procedure PARTITION and all the required preprocessing take $O(dn \log n)$ time.
Proof: For brevity the proof is not included.

□

In practical situations one could execute several variations of the algorithm presented in this paper (e.g., the one in [GZ1]) and then select a solution with least (d-1)-volume. We conjecture that an approximation algorithm based on this approach generates solutions that are very close to optimal. However, proving a smaller bound for this approach seems to be difficult. A postprocessing procedure that transforms E_{apx} into a feasible solution in which each hyperplane segment includes at least one point can be easily constructed. In general it will not generate better solutions, but in many cases the solution generated by our algorithm will be improved.

V. References.

[DC] Du, D. Z. and Chen Y. M., "On Fast Heuristics for Minimum Edge Length Rectangular Partition," Technical Report, MSRI 03618-86, Feb. 1986.

[DPS] Du, D. Z., L. Q. Pan and M. T. Shing, "Minimum Edge Length Guillotine Rectangular Partition," Technical Report, MSRI 02418-86, Jan. 1986.

[GRSZ] Gonzalez, T., M. Razzazi, M. Shing and S. Zheng, "On Optimal d-Guillotine Partitions Approximating Hyperrectangular Partitions," Technical Report TR-89-25, CS Dept., UCSB, October 1989.

[GZ1] Gonzalez, T. and S.-Q. Zheng, "Bounds for Partitioning Rectilinear Polygons", Proc. Symp. Computational Geometry, June 1985, pp. 281-287, (also appears as Technical Report #85-22, CS Dept., UCSB, Dec. 1985).

[GZ2] Gonzalez, T. and S.-Q. Zheng, "Approximation Algorithms for Partitioning Rectilinear Polygons", Technical Report # 85-21, CS Dept., UCSB, Dec. 1985, (to appear in *Algorithmica*).

[GZ3] Gonzalez, T and S. Q. Zheng, "Improved Bounds for Rectangular and Guillotine Partitions," *Journal of Symbolic Computation*, 7, 1989, pp 591 - 610.

[K] Kirkpatrick, D.G., "An Upper Bound for Sorting Integers in Restricted Ranges", Proc. 18th Allerton Conf. on Communication, Control and Computing, Illinois, Oct. 1980.

[L] Lingas, A., "Heuristics For Minimum Edge Length Rectangular Partitions of Rectilinear Figures," 6th GI-Conference, Dortmund, 1983, Lecture Notes in Computer Science. 195 (Springer-Verlag).

[LPRS] Lingas, A., R. Y. Pinter, R. L. Rivest, and A. Shamir, "Minimum Edge Length Partitioning of Rectilinear Polygons," Proc. 20th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, Oct. 1982.

[L1] Levcopoulos, C., "Minimum Length and Thickest-First Rectangular Partitions of Polygons," Proceedings of the 23rd Allerton Conference on Communication, Control and Computing, U. of Illinois, Oct. 1985.

[L2] Levcopoulos, C., "Fast Heuristics for Minimum Length Rectangular Partitions of Polygons," Proceedings of the 2nd Computational Geometry Conference, June 1986.

[R] Rivest, R. L., The "PI" (Placement and Interconnect) System, Proc. 19th Design Automation Conference, June 1982.