Monotone Bipartitioning Problem in a Planar Point Set with Applications to VLSI

PARTHASARATHI DASGUPTA
Indian Institute of Management, Calcutta
PEICHEN PAN
Aplus Design Technologies
SUBHAS C. NANDY
Indian Statistical Institute
and
BHARGAB B. BHATTACHARYA
University of Nebraska-Lincoln

A new problem called monotone bipartitioning of a planar point set is identified which is found to be useful in VLSI layout design. Let F denote a rectangular floor containing a set A of n points. The portion of a straight line formed by two points from the set A is called a line segment. A *monotone increasing path* (MP) in F is a connected and ordered sequence of line segments from the bottom-left corner of F to its top-right corner, such that the slope of each line segment is nonnegative, and each pair of consecutive line segments share a common point of A. An MP is said to be maximal (MMP) if no other point in A can be included in it preserving monotonicity. Let A^L denote the subset of A corresponding to the end points of the line segments in an MMP, L. The path L partitions the set of points $A \backslash A^L$ into two subsets lying on its two sides. The objective of monotone bipartitioning is to find an MMP L, such that the difference in the number of points in these two subsets is minimum. This problem can be formulated as finding a path between two designated vertices of an edge-weighted digraph (the weight of an edge being an integer lying in the range $[-n, n]$), for which the absolute value of the algebraic sum of weights is minimized. An $O(n \times e)$ time algorithm is proposed for this problem, where e denotes the number of edges of the graph determined from the geometry of the point set. The monotone bipartitioning problem has various applications to image processing, facility location, and plant layout problems. A related problem arises while partitioning a VLSI floorplan. Given a floorplan with n rectangular blocks, the goal is to find a monotone staircase channel from one corner of the floor to its diagonally opposite corner such that the difference in the numbers of blocks lying on its two sides is minimum. The problem is referred to as the staircase bipartitioning problem. The proposed algorithm for a point set can be directly used to solve this

Authors' current addresses: P. Dasgupta, Computer Science and Engineering Dept., Electrical and Computer Engineering Dept., University of California, San Diego, 3819 APM, UCSD, San Diego, CA 92093-0114; email: partha@cs.ucsd.edu (on leave from the Indian Institute of Management, Calcutta, India); P. Pan, Aplus Design Technologies, Inc., 10850 Wilshire Blvd., Los Angeles, CA 90024; email: peichen@aplus-dt.com; Subhas C. Nandy, Indian Statistical Institute, 203 B. T. Road, Calcutta 700 108, India; email: nandysc@isical.ac.in; B. B. Bhattacharya, Dept. of CSE, University of Nebraska-Lincoln, Lincoln, NE 68588; email: bhargab@cse.unl.edu (on leave from the Indian Statistical Institute, Calcutta, India).

problem in $O(n^2)$ time. However, an improved $O(n)$ time algorithm is reported for this special case. This leads to an $O(n \log n)$ time algorithm for hierarchical decomposition of a floorplan with a sequence of staircase channels. Staircase bipartitioning has many applications to channel and global routing.

Categories and Subject Descriptors: B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI (very large scale integration)*; B.7.2 [**Integrated Circuits**]: Design Aids—*placement and routing*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; J.2 [**Physical Sciences and Engineering**]: *Engineering*; J.6 [**Computer-Aided Engineering**]: *computer-aided design (CAD)*

General Terms: Theory, Design, Algorithms

Additional Key Words and Phrases: Complexity of algorithms, floorplanning, partitioning, routing, very large scale integration (VLSI)

1. INTRODUCTION

Circuit partitioning is a well-known and important problem in VLSI layout design. A circuit is usually represented by a graph or a hypergraph [Sherwani 1999], where the nodes represent the modules or circuit components, and the edges capture the interconnection information. The goal of partitioning is to decompose the circuit into several components satisfying certain constraints on area, number of terminals, etc., and to optimize some objective function(s), for example, the number of interpartition interconnections, intermodule longest delay, to name a few. Partitioning is required in all levels of the circuit design from the behavioral description to its final layout stage. Several graph bisection techniques are widely used in VLSI placement and floorplan design [Kernighan and Lin 1970; Sherwani 1999].

In this paper, we identify a new partitioning problem called *monotone bipartitioning* of a planar point set. Let F denote a rectangular floor containing a set A of n points that includes the bottom-left corner (s) and the top-right corner (t) of the floor. A *monotone increasing path* (*MP*) through A is a connected and ordered sequence of straight-line segments, say $\{l_1, l_2, \ldots, l_k\}$, such that each l_i is bounded by two points from the set A, and has nonnegative slope. Thus, each pair of consecutive line segments (l_i, l_{i+1}) on the path must share a common element of A. The starting point of l_1 and the end point of l_k are the bottom-left and top-right corners of the floor, respectively. An *MP* in A is said to be a *maximal monotone increasing path* (*MMP*) if no other point in A can be included in it preserving its monotonicity. Consider an *MMP* L, and let the set of points on L be denoted by A^L. The path L in F partitions the set of points $A \backslash A^L$ into two subsets, one on its left and the other on its right; our objective is to find an *MMP* L such that the difference of the number of points in these two subsets is minimum. This can be formulated as a search problem on an edge-weighted directed graph with n nodes. The nodes of the graph correspond to the points in A; the edges and their weights are determined from their positions on the floor. In this formulation, the weight of an edge turns out to be an integer lying in the range $[-n, n]$. Here, the goal is to find a path between the nodes s and t in the graph, such that the absolute value of the sum of edge weights along the path is minimum. An $O(n \times e)$ time algorithm is proposed, where e

denotes the number of edges in the graph. Thus, e may be $O(n^2)$ in the worst case. Monotone bipartitioning using a maximal monotone decreasing path from the top-left corner of the floor to its bottom-right corner can be obtained in a similar fashion.

A VLSI floorplan is a rectangular dissection of a bounding rectangle with isothetic cut-lines [Sherwani 1999]. The bounding rectangle represents the chip floor, each basic rectangle denotes a circuit module or a block, and a cut line denotes routing space (channel). Given a floorplan F with n rectangular blocks, the objective of staircase bipartitioning is to split F by a staircase (monotone increasing/decreasing) channel from one corner of F to its diagonally opposite corner, such that the difference in the numbers of blocks on the two sides of the channel is minimum. The concept of staircase channels was first introduced in Guruswamy and Wong [1988]. This problem can be solved using the algorithm for monotone bipartitioning of point set as introduced earlier, in $O(n^2)$ time since the underlying graph is planar. Exploiting certain geometric properties of a floorplan, we then report an improved $O(n)$ time algorithm for this special case. This leads to an $O(n \log n)$ time algorithm for hierarchical staircase bipartitioning of a floorplan. Such hierarchical decomposition facilitates global routing satisfying safe (acyclic) channel routing order.

The paper is organized as follows. In Section 2, we introduce the monotone bipartitioning problem. Application of this problem to hierarchical floorplan partitioning is described in Section 3. In Section 4, we present a graph-theoretic formulation and a polynomial-time algorithm for point set bipartitioning. In Section 5, a linear-time algorithm for staircase bipartitioning is reported. In Section 6, a technique for hierarchical partitioning of a VLSI floorplan is described. Finally, in the last section, we conclude the paper and suggest a few future research directions.

2. MONOTONE BIPARTITIONING PROBLEM

Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of n points distributed arbitrarily on a two-dimensional rectangular floor F. We assume that a_1 and a_n are, respectively, the bottom-left and top-right corners of the floor. Henceforth, the points a_1 and a_n will alternatively be denoted as s and t, respectively. The coordinate of a point $a_i \in A$ will be denoted as (x_i, y_i). The problem is to partition the plane by a piece-wise linear curve consisting of a set of straight line segments $\{l_1, l_2, \ldots, l_k\}$ from s to t such that (i) the end points of each l_i coincide with two points in A, (ii) the slope of each l_i is nonnegative, (iii) two consecutive segments l_i and l_{i+1} always share a common point in A, and (iv) the starting point of l_1 and end point of l_k are s and t, respectively. Thus, the sequence of points $\{a_{i1}(= s), a_{i2}, \ldots, a_{ik-1}, a_{ik}(= t)\}$ on an MP, say L, satisfies the following *monotone increasing property*:

$$x_{i1} \leq x_{i2} \leq \cdots \leq x_{ik} \quad \text{and} \quad y_{i1} \leq y_{i2} \leq \cdots \leq y_{ik}.$$

The path L formed by the above sequence is called a *monotone increasing path* (MP), and the corresponding partition of F is referred to as a *monotone bipartition*.

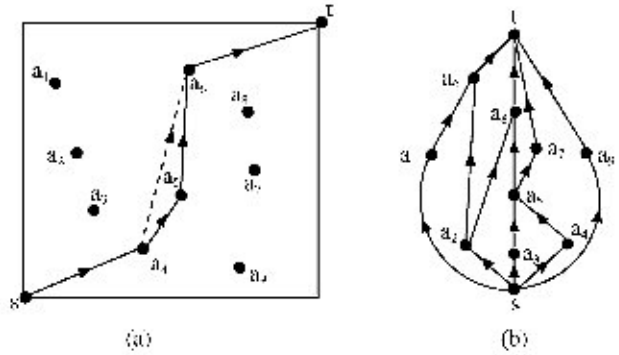

Fig. 1. (a) An example floor containing a set of points, and (b) the transitively reduced digraph G_T.

Definition 1. A monotone increasing path L is said to be a *maximal monotone increasing path* (*MMP*) if the insertion of any other point of A in path P violates the monotone increasing property of P.

Let A^L denote the set of points on an *MMP* L. The path L partitions the set of points $A \backslash A^L$ into two subsets lying on its two sides. The weight (or cost) $\delta(L)$ of L is the difference of the number of points in these two subsets. In other words, if n_L^ℓ and n_L^r denote, respectively, the number of points to the left and right side of L, then $\delta(L)$ is equal to the absolute value of $(n_L^\ell - n_L^r)$. Now we address the following two problems:

P1: Find an *MMP* L such that $\delta(L)$ is minimum among all other *MMP*'s present on the floor.

P2: Let the points in A be of two colors, say red and blue. In this version of the problem, an *MMP* is defined through the red points only. For such an *MMP* L, let $\delta(L)$ denote the difference in the number of blue points on the two sides of L. The objective is same as that of problem P1.

In Figure 1(a), the path $L_1 = s \rightarrow a_4 \rightarrow a_6 \rightarrow t$ is an *MP*, but not an *MMP*, whereas the path $L_2 = s \rightarrow a_4 \rightarrow a_5 \rightarrow a_6 \rightarrow t$ is an *MMP*. Many such *MMP*'s may exist on a floor, and $\delta(L)$ may be any integer in the interval $[0, (n-3)]$. In Figure 1(a), the *MMP* L_2 is a solution of problem P1 with $\delta(L_2) = 0$. In Figure 2, we demonstrate that the minimum value of $\delta(L)$ may be $O(n)$ in the worst case. Problem P1 finds many applications to image processing [Conti et al. 1999], facility location, and plant layout [Schobel 1998]. In the next section, we discuss an important application of problem P2 to VLSI layout design.

3. APPLICATIONS OF MONOTONE BIPARTITIONING

An interesting problem of VLSI layout design is the staircase bipartitioning of a VLSI floorplan [Majumder et al. 1998, 2001]. A floorplan is said to be *slicible* if it is either a single block, or there is a single isothetic cut-line (slice) that partitions the enclosing rectangle into two slicible floorplans (see

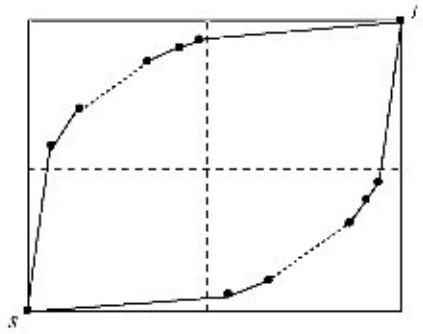

Fig. 2. Example of a point set with optimal partition cost greater than 1.

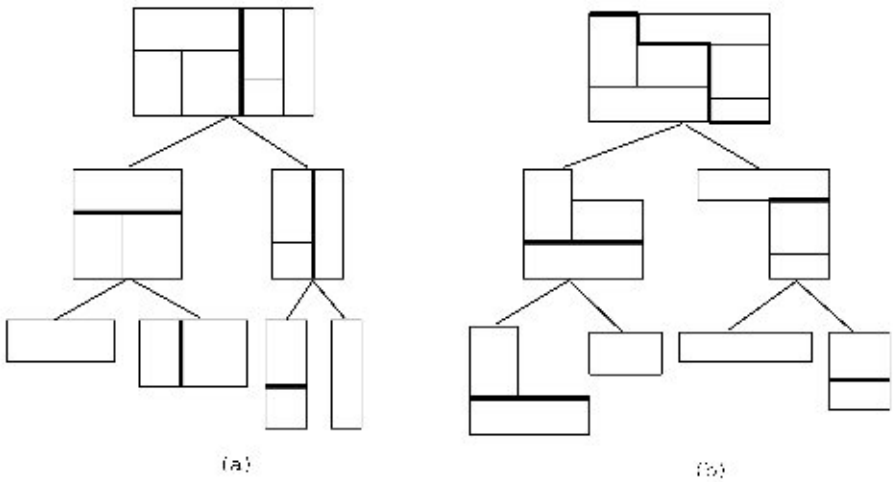

Fig. 3. Hierarchical bipartitioning of (a) a slicible floorplan, (b) a nonslicible floorplan.

Figure 3(a)). Floorplans that are not slicible, are called *nonslicible* floorplans (see Figure 3(b)).

For convenience of wire routing, the channels are routed following a certain order, called *safe (cycle-free) routing order*. A slicible floorplan can be split recursively by the cut-lines to form a *slicing tree* in a top-down fashion, and it is known that a safe routing order always exists among the channels. For such floorplans, particularly those with a balanced slicing tree structure, efficient routing techniques are available [Luk et al. 1987a, 1987b]. For nonslicible floorplans, such a partition using isothetic cut-lines does not exist, which leads to an unfeasible (or unsafe) routing order [Sherwani 1999; Sur-Kolay and Bhattacharya 1991]. Such a routing instance may be handled by introducing certain switchbox regions [Yan and Hsiao 1996]. However, a switchbox is more difficult to route compared to a channel. Alternatively, if the channel definition is generalized to a monotone (increasing/decreasing) staircase, then both the problems of decomposing the floorplan (Figure 3(b)), and that of finding a *safe routing order* [Guruswamy and Wong 1988; Sur-Kolay and Bhattacharya 1991]

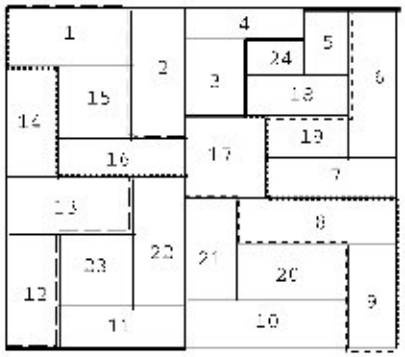

Fig. 4. Hierarchical bipartitioning of a floorplan [Wimer et al. 1989].

can easily be solved. An efficient routing technique for staircase channels has recently been reported in Das et al. [1998].

The design problem discussed above leads to the *staircase bipartitioning problem*: Given a floorplan with rectangular blocks, find a staircase (monotone increasing/decreasing) path through the cut lines from one corner of the floor to its diagonally opposite corner, that partitions the set of blocks into two equal halves. In classical graph partitioning problems, the objective is to find a min-cut bisection; in contrast, the goal here is to find a staircase bisection. This problem can be mapped to an instance of P2, where the T-junctions are the red points, and the center of the blocks are the blue points. Further, the method can be used recursively to partition a general floorplan in a hierarchical manner. An example of a benchmark floorplan and its hierarchical decomposition using staircase channels are shown in Figure 4.

Moreover, since the staircase path is targeted to partition a floorplan into two almost equal halves (balanced) with respect to the number of blocks in each level of recursion, the depth of the hierarchy tends to be the smallest. This may provide certain advantages listed below:

—It is observed that in order to ensure a safe routing order, staircase channels should be routed in a bottom-up fashion [Sur-Kolay and Bhattacharya 1991]. In other words, the channels at the bottom-most level of the hierarchy should be routed first, followed by those lying at the next higher level, and so on. Further, these channels tend to become longer as one moves up from the bottom level to the top level. Thus, balanced bipartitioning offers an efficient divide-and-conquer scheme for global routing.

—In a parallel processing environment, channels belonging to the same hierarchical level can be processed in parallel. Balanced bipartitioning results in a minimum depth of recursion, and in turn, accelerates detailed routing phase [Sherwani 1999] of layout design.

—Staircase bipartition is also applicable to a slicible floorplan, and balancing would lead to an improved routing algorithm.

The monotone bipartitioning problem may find many other important applications to computational geometry, operations research, etc.

4. FORMULATION OF THE PROBLEM

To formulate problem P1, we consider an edge-weighted digraph $G(V, E)$, called monotone graph, where the set of vertices V corresponds to the set of points in A. Thus, $V = \{a_1, a_2, \ldots, a_n\}$, and the set of edges $E = \{(a_i, a_j) | x_i \leq x_j$ and $y_i \leq y_j\}$. The computation of edge weights is described in Subsection 4.1. The digraph G is acyclic. Node s (respectively t) has indegree (respectively outdegree) 0. Any directed path from s to t in G is a monotone path. An important property of the graph G is mentioned in Lemma 1.

Definition 2. [Golumbic 1980] Suppose $\Pi = [\pi_1, \pi_2, \ldots, \pi_n]$ be a permutation of n natural numbers $\{1, 2, \ldots, n\}$. Denote by π_i^{-1} the position of i in Π. The *permutation graph* $G_\Pi(V_\Pi, E_\Pi)$ is an undirected graph with the set of vertices $V_\Pi = \{1, 2, \ldots, n\}$, and the set of edges $E_\Pi = \{(i, j) \mid i, j \in V_\Pi$ and $(i - j)(\pi_i^{-1} - \pi_j^{-1}) < 0\}$.

LEMMA 1. *The undirected version of the digraph G is a permutation graph* [Golumbic 1980].

PROOF. We label the points in A as $1, 2, \ldots, n$, in increasing order of their y-coordinates. Now, a line is swept from the right boundary of the floor to its left boundary, and the labels of the points, as they appear, are recorded. The sequence of labels, thus observed, plays the role of Π. □

We construct the graph G by a plane sweep technique. The points in the set A are processed in increasing order of their y-coordinates. A height-balanced binary tree $\mathcal{T}$ [Cormen et al. 2000] is dynamically maintained on all the points processed so far. When a point a_i is processed, it is inserted in $\mathcal{T}$. Each of the points that appears to the left of a_i in $\mathcal{T}$ should have a directed edge to a_i. The time of construction of the graph G is therefore $O(|E| + n \log n)$.

The digraph G is *transitively orientable*. The transitive reduction of G is denoted as $G_T(V, E_T)$, where $E_T \subseteq E$ is the set of transitively reduced edges of E. The transitive reduction of the graph obtained from the set of points in Figure 1(a) is shown in Figure 1(b). The following lemma is now obvious.

LEMMA 2. *An MMP L on the floor corresponds to a path from s to t in the graph G_T, and vice-versa.*

PROOF. Follows from the construction of the graph G_T. □

4.1 Calculation of Edge Weights

Consider the following problems:

P1: Let (a_i, a_j) be an edge in G_T. Two horizontal lines H_i and H_j are drawn through the points a_i and a_j till they hit the boundary of the floor (see Figure 5). Let n_{ij}^{ℓ} and n_{ij}^{r} be the number of points appearing to the left and to the right of the edge (a_i, a_j) within the horizontal band defined by H_i and H_j. The weight (cost) of the edge (a_i, a_j), denoted by $w(a_i, a_j)$, is equal to $(n_{ij}^{\ell} - n_{ij}^{r})$. Thus, the value of $w(a_i, a_j)$ may be positive, negative, or zero.

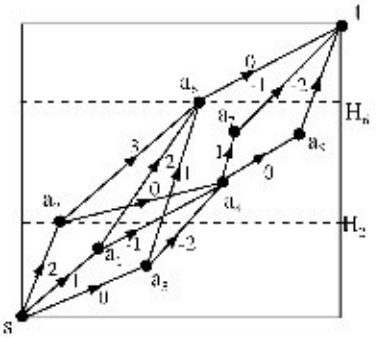

Fig. 5. Computation of edge weights for the monotone bipartitioning problem.

P2: In this case, the floor contains points of two colors, say red and blue. Here G_T is constructed with the red points only. The weight (cost) of an edge is the number of blue points on its left minus the number of blue points on its right, and is calculated as in problem P1.

OBSERVATION 1. *The edge weights for the above two problems may be any integer between $-n$ and n, where n is the total number of points in the set A for problem* P1, *and the number of blue points in the set A for problem* P2.

4.2 An Optimal Solution Method

In both the problems P1 and P2, the number of *MMP*'s may be exponentially large. The following lemma shows that both the problems can be mapped to the problem of finding a path from s to t in the digraph G_T such that the absolute value of the algebraic sum of costs of all the edges on the path is minimum.

LEMMA 3. *Let L be an MMP on the floor, and P be the corresponding path in G_T. Now, $\delta(L)$ is equal to the absolute value of the sum of costs of all the edges in P.*

PROOF. Follows from the definition of the cost of an edge in the digraph G_T. □

Given an acyclic digraph with arbitrary edge cost (any real number with unrestricted sign), the problem of finding a path between two designated nodes such that the absolute value of the sum of edge costs on that path is minimum among all possible paths is known to be NP-hard in Dasgupta et al. [2001]. An efficient heuristic based on depth-first branch-and-bound technique is also proposed in Dasgupta et al. [2001]. However, in the case of point-set bipartitioning, the edge costs have an integer value lying within an interval $[-n, n]$. Lemma 4 indicates an important property of this problem which helps in designing a polynomial-time algorithm for finding the optimal path in the digraph G_T.

Let us consider a vertex $v \in V$, and all possible paths from s to v. Let $D(v)$ denote the set of distinct δ values of those paths. Obviously,

$$D(v) = \bigcup_{v' | (v',v) \in E_T} \bigcup_{d \in D(v')} (d + w(v', v)) \qquad \text{(i)}$$

where $w(v', v)$ is the weight of the edge (v', v).

LEMMA 4. $|D(v)| \leq 2n + 1$.

PROOF. Follows from the fact that number of points that present in one side of the path from s to v may be any integer from 0 to n. □

The algorithm OPTPART, stated below, finds the path of minimum absolute cost from s to t in the weighted digraph G_T, where edge weights satisfy Observation 1.

4.2.1 *Algorithm OPTPART*

Input: The graph G_T, and pointers to the source and target vertices s and t.

Output: A list containing nodes on the optimal path.

Data structure: With each node $v \in V$, an array $D(v)$ of size $2n + 1$, indexed by $[-n, -(n-1), \ldots, -1, 0, 1, \ldots, (n-1), n]$, is attached. Each element of this array is a tuple {*cost*, *back_ptr*}. The *cost*-bit of the d-th element is 1 if there exists a path from s to v with cost d. The *back_ptr* points to the predecessor node v' (of v) which contributes cost d.

In addition, the algorithm maintains a queue Q containing the vertices of the input graph.

```
begin
  set D(s) = {0}; insert(Q, s); (* insert node s in Q *)
  repeat
    v := delete(Q); (* delete the front element of Q and assign it to v *)
    while (v ≠ t) do
      for each node (v*) such that (v, v*) ∈ E_T do
        for each element of D(v) do
        (* Let the d-th element of D(v) be currently under consideration *)
          if the cost-bit of the d-th element is 1 then
              compute d' = d + w(v, v*);
              set cost-bit of d'-th element of D(v*) to 1;
              the back_ptr of the d'-th element points to v;
          endif
        endfor
      endfor
    endwhile
  until (Q is empty);
  (* Retrace the path of maximum absolute cost from t up to s *)
  Let the i-th element be the rightmost element of D(t) containing "1", and
          j-th element be the leftmost element of D(t) containing "1".
  if |j| > i then k = j else k = i endif;
  retrace from the k-th element of D(t) using back_ptr up to node s for
  reporting the optimal path.
end.
```

4.2.2 *Complexity of OPTPART.* In the algorithm OPTPART, when a vertex v is deleted from Q, all its outgoing edges are processed. For each such edge, its cost is added with all elements of $D(v)$ having *cost*-bit set to 1. As the graph is

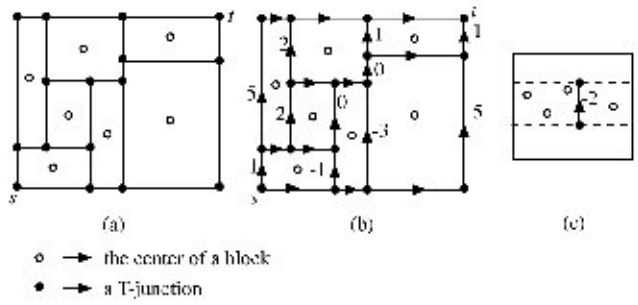

Fig. 6. (a) A floorplan, (b) the weighted floorplan graph, (c) computation of edge weights.

acyclic, if a vertex is deleted from Q, it is never reinserted in Q. Eventually, we are considering all the edges in E_T, and for each edge (v, v^*), its cost is added to at most n values of $D(v^*)$. So, the time complexity of this pass is $O(n|E_T|)$. The time requirement for reporting the optimal path is $O(n)$ in the worst case. Thus overall time complexity of this algorithm is $O(n|E_T|)$, where $|E_T|$ may be $O(n^2)$ in the worst case.

5. STAIRCASE BIPARTITIONING OF A FLOORPLAN

We now present a bipartitioning scheme for a VLSI floorplan with n rectangular blocks using a staircase channel such that the number of blocks in each part becomes equal to at least $\lfloor \frac{n}{2} \rfloor$.

5.1 Staircase Bipartitioning Problem

Consider a VLSI floorplan containing a set of rectangular blocks. Initially, we assume that the boundary of the floorplan is rectangular. We show that the optimal staircase bipartitioning problem can directly be mapped to Problem P2. In this case, corner nodes s and t, and all T-junctions of the floorplan are considered as red points; the centers of the blocks are treated as blue points. The underlying planar graph, called a *floorplan graph*, is an edge-weighted digraph whose vertices correspond to the set of red points. A directed edge e_{ij} is placed from v_i to v_j if the corresponding T-junctions are consecutive along the boundary of a block, and v_j appears either to the right or above v_i in the floorplan. Thus, all the edges of the floorplan graph can be embedded either horizontally or vertically. The weight of each vertical edge e_{ij} is now assigned as in Problem P2. The weight of each horizontal edge is set to 0. An example of a floorplan and its corresponding graph are shown in Figures 6(a) and 6(b), respectively. The computation of edge weights is illustrated in Figure 6(c). The following lemma now leads to a solution strategy for the staircase bipartitioning problem:

LEMMA 5. *Each directed path from s to t in the floorplan graph is an MMP, and corresponds to a staircase bipartition of the floorplan and vice-versa.*

PROOF. Follows from the fact that no transitive edge is present in the floorplan graph. □

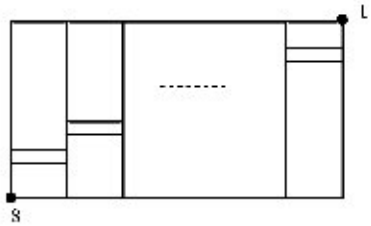

Fig. 7. A floorplan with exponential number of paths.

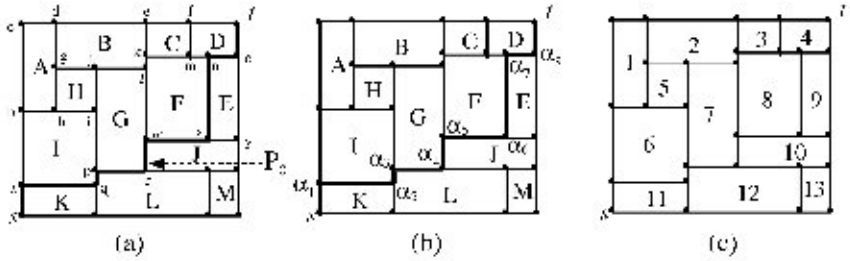

Fig. 8. Labeling blocks in a floorplan.

The number of maximal monotone paths from s to t in a floorplan can be exponentially large, as shown in Figure 7. In order to find an optimal staircase bipartition, Algorithm OPTPART of Section 4.2.1 may be used directly in this context. Thus, the time complexity would be $O(n^2)$ for a floorplan with n blocks, as the underlying graph is planar. In the next subsection, we explore a few additional characteristics of the floorplan graph which lead to a simple $O(n)$ time algorithm for this problem.

5.2 An Improved Method for Staircase Bipartitioning

In this method, the floorplan graph is assumed to be unweighted. Its vertices are classified into three types, namely *type-0*, *type-1*, or *type-2*, depending on whether its outdegree is 0, 1, or 2. Among the four corners of the floorplan, vertex t is of *type-0*, vertices corresponding to the bottom-right and the top-left corners are of *type-1*, and vertex s is of *type-2*. The vertices corresponding to the T-junctions are either of *type-1* or of *type-2*.

Definition 3. A block $\mathcal{B}$ is said to be attached to a node v of the floorplan graph if the bottom-left corner of block $\mathcal{B}$ coincides with the T-junction corresponding to node v in the floorplan graph.

Definition 4. If the top and left boundaries of a block are fully spanned by a sequence of consecutive edges of a staircase path P in the floorplan, the block is said to be bounded by P.

For example, consider the floorplan in Figure 8(a). A staircase path $P_0 = \{s, a, q, p, r, w, x, n, o, t\}$ is indicated by dark lines. By Definition 4, blocks K and E are bounded by P_0.

LEMMA 6. *Every staircase path in a floorplan must have at least one block bounded by it.*

PROOF. [By contradiction] Let us assume that a staircase path, say P_0, exists by which no block in the floorplan is bounded. The vertices along P_0 are marked as $\alpha_1, \alpha_2, \ldots, \alpha_k$ (k is the number of vertices on P_0 including s and t) (see Figure 8(b)). If the initial edge (starting at s) is vertical, then the edge (α_1, α_2) completely spans the left boundary of the block (say K in Figure 8(b)), which is placed at the bottom-left corner of the floor. By our assumption of contradiction, vertex α_2 must appear at any position on the top boundary of block K excluding its top-right corner. Since the path is monotone and cannot penetrate any block, it will again completely span the left boundary of the block whose bottom-left corner is at α_2. The progress of the path continues in this manner until a vertex on the top boundary of the floor is reached, which is the top-left corner of the block (say D in Figure 8(b)) placed at the top-right corner of the floor. The next horizontal movement up to vertex t will span the top boundary of block D. Since the left boundary of block D is already assumed to be spanned by P_0, we infer that block D is bounded by P_0; otherwise, a block like E will be bounded by P_0. This contradicts the assumption. Similarly, it can be proved that if the initial edge of P_0 is horizontal, at least one block is bounded on the floor. □

5.2.1 *Block Labeling Method.* We now label the blocks of the floorplan in an appropriate manner that leads to an optimum solution of staircase bipartitioning.

We consider a geometric embedding of the floorplan graph. Each *type-2* vertex has two successors, one along vertical direction (toward the top) and the other along horizontal direction (toward the right). They are referred to as *left successor* and *right successor*, respectively. We traverse the graph in a manner similar to depth-first traversal, starting from vertex s, until all the edges are visited. During traversal, if a *type-2* vertex is reached, its left successor is explored before the right successor. As soon as t is reached during traversal, a staircase path is identified. By Lemma 6, at least one block must be bounded by that path. We label that block as $\lambda + 1$, where λ is the number of blocks labeled so far. If more than one block qualify, we consider the latest one visited during forward traversal. In order to explore another staircase path in a systematic manner, we backtrack until a vertex u is reached whose one successor is unexplored.

OBSERVATION 2. *Let v denote the vertex where backtracking terminates. Then (i) v is a type-2 vertex, and (ii) the block attached to v is unlabeled, and bounded by the current staircase path. This block is the candidate that receives the label $\lambda + 1$.*

In order to label all the blocks, we may generate other staircase paths. However, identification of all such paths may require time exponential in the number of blocks. On the contrary, since our goal is to label the blocks, we modify the depth-first traversal such that each edge of the floorplan graph is visited exactly

once. In order to prevent multiple visits of an edge, we attach a *tag* bit to each edge which is initialized to 0. When an edge is visited once, its *tag* bit is set to 1. The new traversal procedure is described below:

During the forward traversal, if t is reached or all the outgoing edges of the current vertex v are already visited, the forward traversal stops. If $v \neq t$ then v must be a *type-1* vertex. The staircase path may be constructed by concatenating a path from v to t (explored earlier) with the currently explored path from s to v. We start backtracking from v until the vertex u (whose right successor is unexplored) is reached. The block attached to u is labeled and forward traversal is resumed from u toward its right successor. The process terminates when s is reached during backtrack and *tag* bits of its two outgoing edges are set.

Algorithm **Block_Labeling**

Input: The floorplan graph; a list of vertices corresponding to the T-junctions. Each vertex is attached with the following:

—Two pointers, called *left_successor* and *right_successor*. If a vertex is of *type-1*, its *right_successor* = NULL. For the vertex t, both the pointers are NULL.

—Two tag bits, called *left_tag* and *right tag*, initialized to 0.

—The block attached to that vertex.

Output: A labeling of all the blocks of the floorplan.

Additional data structures: A stack S for traversal of the floorplan graph.

```
begin
  (* Initialize *) λ ← 1; current_vertex ← s;
  (* Depth-first traversal *)
  while not both the tag-bits of s is equal to 1 do
    (* Proceed forward *)
    if current_vertex ≠ t and both left_tag and right_tag of current_vertex
      are not equal to 1 then
      if (left_tag of current_vertex ≠ 1) then (* proceed left *)
          PUSH(current_vertex) (* in stack S *);
          next_vertex ← left_successor;
          left_tag ← 1;
      else (* proceed right *)
          PUSH current_vertex in stack S;
          next_vertex ← right_successor;
          right_tag ← 1
      endif
    (* Backtrack *)
      while ((right_successor of current_vertex = NULL)
          or (right_tag of current_vertex = 1)) do
          current_vertex ← POP(S); (* POP element from stack *)
      endwhile
```

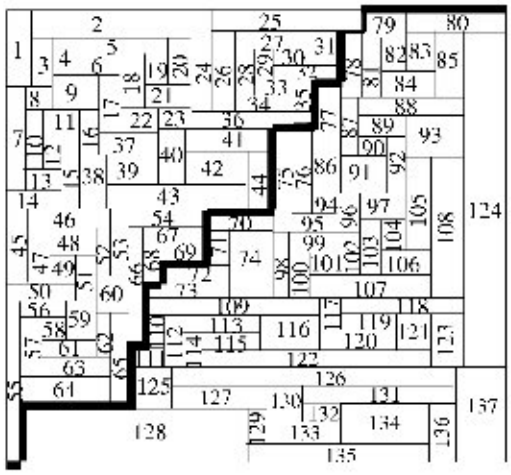

Fig. 9. Bipartition of the benchmark example of [Cai and Wong 1990].

```
        (* Label a block *)
            if (right_tag of current_vertex = 0) then
                    Label the block attached to the current_vertex by λ;
                    λ ← λ + 1;
            endif
        endwhile
end.
```

LEMMA 7. *The proposed Block_Labeling algorithm labels each block exactly once, and the labels of all the blocks are distinct.*

PROOF. During the traversal, a block is labeled when its attached vertex is reached for processing its right successor. The first part of the lemma follows from the fact that each edge is processed only once. The uniqueness of the labeling follows from the uniqueness of the traversal sequence of edges in a depth-first fashion. □

We demonstrate our algorithm by labeling the blocks of the floorplan of Figure 8(a). The algorithm starts at node s, and proceeds via a, b, c, d, e, and f up to the vertex t. Then it backtracks up to the *type-2* vertex b, and labels block A as 1. Next, it resumes forward traversal from vertex b, and after visiting h and g, reaches at d. Since the outgoing edge of d is already visited, backtracking is continued up to the *type-2* vertex g whose one successor is yet to be visited. The block B is labeled as 2. This method is continued to label all the blocks. The final labeling is shown in Figure 8(c). The algorithm has been implemented and tested on several benchmark floorplans. Figure 9 shows the block labeling in a benchmark floorplan [Cai and Wong 1990].

LEMMA 8. *The time complexity of the proposed Block_Labeling algorithm is $O(n)$.*

PROOF. Given the floorplan, its graph representation can be constructed in $O(n)$ time. Each edge of the floorplan graph is visited only twice (once in the forward pass and once during backtrack). Since in a planar graph the number of edges is $O(n)$, the lemma follows. □

5.2.2 *Optimal Bipartitioning.* The above block labeling method can now be used to find a staircase that partitions the floor such that the blocks in the two partitions differ in count by at most 1.

LEMMA 9. *Let σ be a staircase path partitioning the set of blocks on the floor to its left and right sides, respectively. Let λ_i^{max} and λ_i^{min} be the maximum and minimum labels among all blocks in the i-th side of σ, i = left and right. Now,*

(i) $\lambda_{left}^{max} \leq \lambda_{right}^{min}$, *and*
(ii) *both the blocks having labels λ_{left}^{max} and λ_{right}^{min} are aligned with the path σ (to its left and right, respectively).*

PROOF. The first part follows from the order of processing of outgoing edges of *type-2* vertices. The second part trivially follows from the first part of this lemma. □

LEMMA 10. *For a given k $(0 \leq k \leq n)$, there exists at least one staircase path with k blocks to its left.*

PROOF. [By induction] Let us assume that a staircase path σ exists which has k blocks labeled as $1, 2, \ldots, k$, on its left. The claim is true for $k = 0, 1$. We show that there exists a staircase σ' with $k + 1$ blocks to its left, which are labeled as $1, 2, \ldots, k + 1$ according to our labeling scheme. After labeling a block with label k, we resume forward traversal followed by a backtrack. Let the backtrack terminate at a *type-2* vertex u whose right successor is unexplored. The block whose bottom-left corner is u, is labeled as $k + 1$. Next, forward traversal is resumed from u toward t, through the right successor of u. This defines a new staircase path σ' as mentioned in Subsection 5.2.1. Thus, the newly labeled block remains to the left of σ'. We now need to prove that the blocks labeled $0, 1, \ldots, k$ still remain to the left side of σ'. Forward traversal from u terminates as soon as we arrive at a vertex v which is already visited. Thus, σ' is obtained from σ by altering its previous path from u to v by the current path. Thus, all the blocks that appeared on the left side of σ will still remain on the left of σ'. Hence the lemma follows. □

THEOREM 1. *The time complexity of recognizing the staircase path, such that the number of blocks lying on its two sides is $\lfloor \frac{n}{2} \rfloor$ each, is $O(n)$.*

PROOF. The existence of such a staircase follows from Lemma 10. We go on labeling the blocks until a block is at least labeled as $\lfloor \frac{n}{2} \rfloor$. The corresponding staircase is the desired solution. The time complexity follows from Lemma 8. □

6. HIERARCHICAL BIPARTITIONING

The proposed bipartitioning scheme for a rectangular floorplan can easily be tailored to handle any arbitrary orthoconvex floorplan, as described below. This

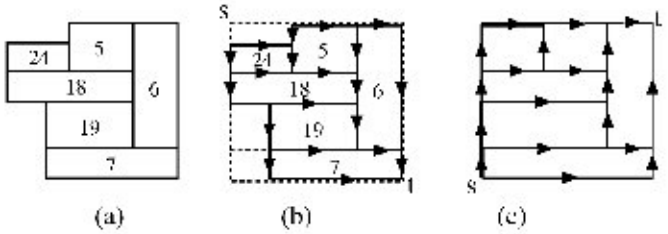

Fig. 10. (a) An orthoconvex subfloorplan, (b) illustration of *free* edges and direction of edges before preprocessing, and (c) floorplan graph of a subfloorplan after preprocessing.

leads to hierarchical partitioning of a VLSI floorplan using a sequence of staircase channels.

Definition 5. An isothetic polygon (a polygon whose sides are parallel to the coordinate axes) is said to be *orthoconvex* if any horizontal or vertical line, not aligned with its boundaries, intersects the polygon at exactly two points or at no point [Rawlins and Wood 1988].

When a rectangular floorplan is split into two parts by a staircase channel, its two components and those appearing in the subsequent levels of the hierarchy may no longer be rectangular in shape. However, it is easy to prove that each of them will look like an isothetic orthoconvex polygon. Further, it is often desirable to have staircase channels in two consecutive levels of the hierarchy with reversed orientations (i.e., if a channel in the ith level is an increasing staircase, then that in the $(i + 1)$th level should be a decreasing staircase).

We now describe a preprocessing procedure to take care of the following steps: (i) mapping a nonrectangular orthoconvex floorplan to a rectangular one, (ii) redefining the two designated nodes s and t, and (iii) modifying the floorplan graph to determine a staircase of reversed orientation.

6.1 Rectangularization of an Orthoconvex Floorplan

Let $\mathcal{F}$ be a nonrectangular orthoconvex floorplan. We first find the smallest isothetic rectangle $\mathcal{R}$ containing $\mathcal{F}$. At least four boundary edges of $\mathcal{F}$ will coincide with the four sides of $\mathcal{R}$. Among the other boundary edges of $\mathcal{F}$, those which span a block completely, are called *free edges*.

For each free edge, we inflate the corresponding block toward the boundary of $\mathcal{R}$ to make $\mathcal{F}$ rectangular. For an illustration of the procedure, consider the floorplan in Figure 10(a), which is nonrectangular and orthoconvex. This is obtained by two successive bipartitions of the floorplan shown in Figure 4. In Figure 10(b), the corresponding smallest enclosing rectangle $\mathcal{R}$ is shown by dotted lines, and the *free edges* are highlighted with dark lines. In Figure 10(c), the modified rectangular floorplan is shown.

6.2 Algorithm and Complexity

Given a nonrectangular floorplan, we convert it to a rectangular floorplan as described in the earlier subsection. We choose s and t based on the orientation

of the staircase in the previous level of the hierarchy. If the earlier one is an increasing (respectively decreasing) staircase, then the one at this level should preferably be of decreasing (respectively increasing) orientation. At each level, we therefore need to change the directions of the edges in the floorplan graph inherited from the previous level. See Figures 10(b) and 10(c) for illustrations. The optimal staircase channel in the modified floorplan $\mathcal{R}$ is then determined. This provides a solution for the original floorplan $\mathcal{F}$.

The aforesaid procedure can be recursively used for a complete hierarchical bipartition of a floorplan.

THEOREM 2. *The worst-case time complexity of the proposed scheme of hierarchical bipartitioning of a floorplan is* $O(n \log n)$.

PROOF. By Theorem 1, there exists a staircase which partitions a floorplan with k blocks, such that each part contains at least $\lfloor \frac{k}{2} \rfloor$ blocks. Thus, the number of levels in the hierarchy is $O(\log n)$. At each level, construction of all floorplan graphs and finding staircase channels requires a total of $O(n)$ time. Hence the proof follows. □

7. CONCLUSION

A new problem called monotone bipartitioning of a planar point set is introduced in this paper. A graph-theoretic formulation of the problem and an $O(ne)$ time algorithm is proposed, where n is the number of points in the set and e is the number of edges in the derived graph. A related problem is to partition a VLSI floorplan using a staircase channel such that the blocks on the two sides of the channel differ in count by at most 1. An efficient linear time algorithm for this problem is proposed. This leads to an $O(n \log n)$ time algorithm for hierarchical decomposition of a VLSI floorplan satisfying safe routing order.

A more practical problem in the context of VLSI floorplan partitioning is to divide the floor into two parts of almost equal numbers of blocks and simultaneously minimizing the number of crossing nets. Thus, the goal is to search for a path χ which minimizes the objective function $cost(\chi) = \theta \times \delta(\chi) + (1-\theta) \times f(\chi)$, where $\delta(\chi)$ is the difference in the number of blocks on the two sides of the cut and $f(\chi)$ is the number of nets crossing the cut. The parameter θ may be chosen by the designer depending on the importance of the two factors participating in the objective function. For $\theta = 0$, the problem reduces to min-cost staircase partitioning, which is shown to be solvable in polynomial time [Majumder et al. 1988]. For $\theta = 1$, it reduces to the problem discussed in this paper, which is solvable in $O(n)$ time. Design of an efficient algorithm for the general problem is currently under investigation.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their constructive comments. One referee indicated that the point set bipartitioning problem may find further applications to domain decomposition for solving partial differential equations in parallel.

REFERENCES

CAI, Y. AND WONG, D. F. 1990. A channel/switchbox definition algorithm for building block layout. In *Proceedings 27-th ACM/IEEE Design Automation Conference* (DAC), (June 1990), 638–641.

CONTI, F., MALUCELLI, F., NICOLOSO, S., AND SIMEONE, B. 1999. On a 2-dimensional equipartition problem. *European J. Operat. Res. 113*, 215–231.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2000. *Introduction to Algorithms*, 2nd ed. MIT Press, Cambridge, MA.

DAS, S., SUR-KOLAY, S., AND BHATTACHARYA, B. B. 1998. Routing of L-shaped channels, switchboxes and staircases in manhattan-diagonal model. In *Proceedings 11th International Conference on VLSI Design* (Jan. 1998), 65–70.

DASGUPTA, P., SEN, A. K., NANDY, S. C., AND BHATTACHARYA, B. B. 2001. Searching networks with unrestricted edge costs. *IEEE Trans. Syst., Man Cybern. A 31*, 6 (Nov.), 497–507.

GOLUMBIC, M. C. 1980. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, NY.

GURUSWAMY, M. AND WONG, D. F. 1988. Channel routing order for building block layout with rectilinear modules. In *Proceedings of the IEEE/ACM International Conference on CAD (ICCAD)*. 184–187.

KERNIGHAN, B. W. AND LIN, S. 1970. An efficient heuristic procedure for partitioning graphs. *Bell Sys. Tech. J. 49*, 2, 291–307.

LUK, W. K., SIPALA, P., TAMMINEN, M., TANG, D., WOO, L. S., AND WONG, C. K. 1987a. A hierarchical global wiring algorithm for custom chip design. *IEEE Trans. CAD 6*, 4, 518–532.

LUK, W. K., SIPALA, P., AND WONG, C. K. 1987b. Minimum-area wiring for slicing structures. *IEEE Trans. Comput. 36*, 6, 745–760.

MAJUMDER, S., NANDY, S. C., AND BHATTACHARYA, B. B. 1998. Partitioning VLSI floorplans by staircase channels for global routing. In *Proceedings of the 11th International Conference on VLSI Design* (Jan. 1998). 59–64.

MAJUMDER, S., SUR-KOLAY, S., BHATTACHARYA, B. B., AND NANDY, S. C. 2001. Area (number)-balanced hierarchy of staircase channels with minimum crossing nets. In *Proceedings of the International Symposium on Circuits and Systems* (ISCAS), vol. 5, 395–398.

RAWLINS, G. J. E. AND WOOD, D. 1988. Orthoconvexity and its generalizations. In *Computational Morphology*, Toussaint, G. (Ed.), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 137–152.

SCHOBEL, A. 1998. Locating least-distant lines in the plane. *European J. Operat. Res. 106*, 152–159.

SHERWANI, N. 1999. *Algorithms for VLSI Physical Design Automation*, 3rd ed. Kluwer Academic Publishers, Boston, MA.

SUR-KOLAY, S. AND BHATTACHARYA, B. B. 1991. The cycle structure of channel graphs in nonslicible floorplans and a unified algorithm for feasible routing order. In *Proceedings IEEE International Conference on Computer Design* (ICCD), (1991). 524–529.

WIMER, S., KOREN, I., AND CEDERBAUM, I. 1989. Optimal aspect ratios of building blocks in VLSI. *IEEE Trans. CAD 8*, 2, 139–145.

YAN, J.-T. AND HSIAO, P.-Y. 1996. Minimizing the number of switchboxes for region definition and ordering assignment. *IEEE Trans. CAD 18*, 3, 336–347.

Received June 1999; accepted November 2000