An Innovative Steiner Tree Based Approach for Polygon Partitioning

Yongqiang Lu, Qing Su and Jamil Kawa
Advanced Technology Group, Synopsys Inc.

Abstract— As device technology continues to scale past 65nm, the heavy application of resolution enhancement techniques (RET) makes the complexity, run time and quality issues in mask data preparation (MDP) grow severely. As one major and core step in MDP, polygon partitioning converts the complex layout shapes into trapezoids suitable for mask writing. The partitioning run time and quality of the resulting polygon partitions directly impacts the cost, integrity, and quality of the written mask. In this work, we introduce an innovative approach to solve the polygon partition problem by constructing a variant Steiner minimal tree: minimal partition tree (MPT). We prove the equivalence between MPT and the optimal polygon partition. Also, the solution search space for MPT is further reduced for the efficiency of the MPT algorithms. Finally, a generic MPT algorithm flow and a linear-time heuristic algorithm based on it are proposed. Experiments show that MPT solves the polygon partitioning with very promising and high quality results.

I. Introduction

The application of Steiner tree based algorithms have been widely used in many EDA areas, especially in physical design. Researches have accomplished many efficient heuristic and exact solutions, [1–4]. In this paper, we apply Steiner minimal tree approach to another EDA field, Mask Data Preparation (MDP), to solve the polygon partitioning problem.

VLSI masks are written by electron beams with either one of the two mask writing varieties: Raster mask or variable shaped beam (VSB) mask. One core step for mask data preparation (MDP) is to fracture the polygons of the data of a design into either rectangles or parallel-axis trapezoids in the most general case. With the VLSI technology scaling down deep into the nano meter era, complexity and quality issues for MDP have grown dramatically in severity. This is due to the ever decreasing dimension of the layout objects and to the application of more aggressive RET. Therefore, guaranteeing high quality polygon partition results for modern designs is becoming increasingly difficult.

Traditionally, people applied cut line based heuristics to solve the polygon partitioning problem [5–7]. The use of these heuristics implies going through a cycle of local cut line evaluation, correction, and re-evaluation. However, the ever-increasing demand for a high quality of partitions dictates a big improvement from current algorithms. For example, the main criteria for a high quality polygon partition are: to minimize the number of small unprintable geometries known as slivers, to minimize the exposed boundary length of such slivers, and to avoid CD (critical dimension) slicing [8]. Meeting all those criteria in current cut line based heuristics is becoming harder and the algorithms used are frequently trapped in achieving local optima. Additionally, each cut line based heuristic is designed and tailored toward a specific optimization objective such as minimizing the cut line length, or minimizing the figure count, and so on. If the need to change one or more objectives arises the whole algorithm has to be completely changed. The flexibility and portability to such changes of objective for classical algorithms are low. Furthermore, memory demands and runtime limitations imposes restrictions on the use of more powerful cut line based standard algorithms.

In this paper, we propose an innovative approach for solving the polygon partitioning problem by formulating a variant Steiner minimal tree which is referred to from here on as the minimal partition tree (MPT). Comparing MPT to traditional approaches, this new approach offers a global and systematic algorithm for obtaining an optimal solution. Besides, since MPT inherits a solid theoretical foundation from existing Steiner tree researches, there are several mature and efficient algorithms available that can be applied to solve the reformulated Steiner tree problem. Furthermore, in our generic MPT optimization framework, changing optimization objectives is an easy task and does not imply changing algorithms. An example would be that if the objective is changed from minimizing the total cut length to minimizing the maximum cut length, the variant Steiner tree can simply be switched from a Steiner minimal tree to a min-max Steiner tree. In summary, the key contributions of the proposed work are:

1. A new approach for solving the polygon partitioning problem is introduced by formulating a variant Steiner minimal tree, MPT.
2. A generic algorithm and flow for solving MPT and a linear time heuristic based on it is proposed.
3. A guidance for reducing the size of the solution search space of to improve the efficiency of the new Steiner tree problem is provided.
4. The theoretic and algorithmic foundation of the proposed approach may serve as the basis of applying other Steiner tree algorithms to the polygon partitioning problem.

The paper is organized as follows. In Section II, we briefly introduce some preliminaries related to the rectangular polygon

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

partitioning problem. In Section III, a variant Steiner tree is formulated to solve the rectangular polygon partitioning problem. In Section IV, the solution search space for the MPT problem is analyzed and its size is reduced. A heuristic algorithm for solving MPT is introduced in details in Section V. Then in Section VI experimental results are shown to demonstrate the quality of partition provided by the proposed algorithm. Finally, we summarize our work in Section VII.

II. Preliminaries

In most designs the vast majority of the layout objects have rectilinear boundaries. Hence the rectilinear polygon partitioning problem is the main focus of this work. In this paper, for simplicity of presentation, the word "polygon" always refers to "rectilinear polygon". The proposed work in this paper can be applied to any rectilinear polygon, with or without holes. To keep the illustrations simple we use rectilinear no-hole polygon only. Extension to the "with-hole" case is straightforward.

A. Terminologies of Polygon Partitioning

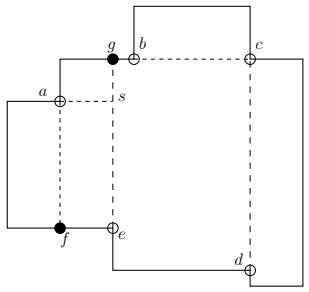

Fig. 1. Partitioning on rectilinear polygon. The dashed lines partition the polygon into a certain number of rectangles.

The terms introduced in this section are all illustrated in Figure 1. The polygon in the figure is a **rectilinear polygon**, which is defined to be a polygon with all the boundaries being axis parallel. An **inflection vertex** (or **I-vertex** for short) of a rectilinear polygon is a concave vertex where its internal angle is 270-degree, such as the points a, b, c, d and e in the figure. **Rectangular partitioning** on the polygon is to divide the polygon into a set of disjoint rectangles whose union equals to the polygon, such as the rectangles separated by the dashed lines in the figure. The **cut lines** are the maximum line segments across the polygon interior that divide the polygon into rectangles. The endpoints of a cut line can only be either on boundaries or on other cut lines. In Figure 1, $\bar{af}$, $\bar{as}$, $\bar{eg}$, $\bar{bc}$ and $\bar{cd}$ are examples of cut lines; while $\bar{gs}$ and $\bar{es}$ are not because they are not the maximum line segments. If a cut line has both I-vertex endpoints then it is called a **chord**, such as $\bar{bc}$ and $\bar{cd}$. Otherwise the cut line is a **cutting ray**, such as $\bar{af}$, $\bar{as}$ and $\bar{eg}$. Draw cutting rays from every I-vertex in two directions toward the polygon interior; they will intersect the closest polygon boundary. The intersection points are called **ray crossing vertices** or **R-vertices** for short, such as the points f and g in the figure. Therefore, every inflection vertex has two corresponding R-vertices: one generated by a horizontal cut ray and one by a vertical cut ray. Each inflection vertex is called the **parent I-vertex** of the two R-vertices generated from it.

In polygon partitioning, the cut lines rather than the rectangles are critical components for problem formulation. All the cut lines, the endpoints and the intersecting points of these cut lines form a graph which is called a **partition graph** (or **partition** in short). The **cost of a partition** is generally measured by the total cut line length. If the graph has no cycles, the corresponding partition is referred to as a **acyclic partition**; otherwise the graph-corresponding partition is referred to as a **cyclic partition**. More particularly, an **anchored partition** is defined as a partition with each of its cut lines having at least one endpoint being an inflection vertex.

B. MCLRP: Minimum Cut Length Rectangular Partition

A general used objective for rectangular partitioning is to make the final partitioned rectangles as close to squares as possible [9]. In other words, the objective is to have the aspect ratio of the partitioned rectangles as close to one as possible. This objective is equivalent to minimizing the total cut line length [9] of a partition, which is traditionally formulated as **minimum cut length rectangular partition (MCLRP)** problem. This is the optimization problem to be solved in this paper.

The MCLRP for a given polygon is not necessarily unique. In [10], it is proved that there always exist an MCLRP as an anchored partition, i.e. a partition with all cut lines anchored on inflection vertices. Hence without any loss of generality we only need to focus on the anchored MCLRP. Since an MCLRP has the minimal total cut line length among all the possible partitions, its cut length is minimal among all the anchored partitions as well. Therefore, it is sufficient to study only the set of anchored partitions for the optimal anchored MCLRP. For simplicity, throughout this paper, an MCLRP refers to a anchored MCLRP and a partition refers to an anchored rectangular partition. The anchored MCLRP has an important property stated in the following Theorem which will be used in subsequent sections.

Theorem II.1 *In a polygon's MCLRP, each inflection vertex of the polygon must have one and only one cut line, unless it has two chords.*

Proof. In an MCLRP of a polygon, if there are two cut lines originated from the same inflection vertex, at least one of them must be movable unless the two cut lines are both chords. A movable edge in a partition is by definition able to be moved by a non-zero distance toward either side in the orthogonal directions without affecting the validity of the partition. For example, in Figure 1, $\bar{as}$ or $\bar{af}$ is movable; while $\bar{cb}$ and $\bar{cd}$ are

not movable because they are chords. The movable edge can be moved until it overlaps with the polygon boundary or another cut line, which will lead to a reduction in the total cut line length. This contradicts with the definition of an MCLRP. □

The statement above gives us an introductory grasp for the rectangular partitioning problem. In the next section we will present a novel approach to solve this problem. The approach is based on a variant Steiner tree formulation.

III. A Variant Steiner Minimal Tree: Minimal Partition Tree (MPT)

The Steiner tree problem has been intensively studied for decades and there have been many efficient and mature algorithms proposed [1–3] with good suboptimal solutions and fast runtime approximations (sub quadratic complexity). In Steiner tree problem, rectilinear Steiner minimal tree (SMT) is used to minimize the total Manhattan wire length connecting the tree terminals. In the partitioning problem, MCLRP solution should minimize the total cut line length. These two problems are very closely related. Solving the MCLRP problem with Steiner tree based approach can bring in many mature and efficient algorithms from Steiner tree researches to solve the partitioning problem. More important, Steiner tree approach enables us to optimize in global view, and avoid re-evaluation of the tree edges again and again. In addition, the Steiner tree approach allows easy change to other objectives. For example, if the objective of partitioning is changed from MCLRP to minimizing the maximum cut length rather than total cut line length, the variant Steiner tree can be switched to be based on min-max Steiner tree. Furthermore, The Steiner tree based algorithms have clear data structure and presentation, hence the implementation is easier and the algorithms can be made very efficient. Motivated by the those benefits, we formulate a variant of Steiner minimal tree, denoted as MPT, to solve the MCLRP problem.

A. Formulation of MPT

The tree to be formulated is constructed on the **Hana grid on a polygon (HGP)**, which is defined as the Hana grids [11] originated from all the polygon vertices and bounded by the polygon. Hence both the portions of the grid lines that are completely interior to the polygon and the polygon boundary lines are included in HGP. The edges that are not overlapped with polygon boundaries are referred to as **cut-state edges** or **edges in cut-state**.

Since MCLRP has all the cut lines anchored on polygon inflection vertices, it follows that all its cut lines lie on the above defined HGP. Therefore, it is sufficient to search through all the partitions on HGP for MCLRP. The variant Steiner tree to solve the MCLRP problem is formulated as an MPT.

Definition III.1 Minimal Partition Tree Problem (MPT)*: Given a rectilinear polygon, it has the I-vertex set I and R-vertex set R. Using the points in $I \cup R$ as terminals we build a rectilinear Steiner tree on the HGP of the polygon connecting these I-vertices and R-vertices subject to the special cost formulation and additional constraints listed as follows:*

1. *The tree edge cost between any two connected vertices is defined as the non-boundary portion of the rectilinear distance between the two points. That is, the tree edge overlapping with the polygon boundary has a zero-cost;*
2. *The tree has no L-shapes (which is defined as a group of two orthogonal tree edges connected by a degree-2 vertex);*
3. *Every I-vertex must have at least one edge in cut-state.*

The above formulated rectilinear Steiner tree is defined as a **partition tree (PT)**. *The rectilinear Steiner minimal tree in the set of PTs is the* **minimal partition tree (MPT)**.

B. Equivalence between MPT and MCLRP

In this subsection, we will prove that the above formulated MPT is equivalent to MCLRP on HGP if the MCLRP is an acyclic partition. Hence solving the MPT problem can lead us the optimal partition solution MCLRP that we need. Toward the end of the subsection, we will discuss solutions to the other cases when the MCLRP is cyclic type.

Lemma III.1 *A partition tree (PT) on the HGP of a given polygon is equivalent to an acyclic partition on the HGP of the polygon.*

Proof. ($\Longrightarrow$) **A PT corresponds to an acyclic partition on HGP with same cost.** From the formulation of PT defined in Definition III.1, all the tree edges of PT are either completely inside the given polygon or on the polygon boundary. The PT edges that are completely inside the polygon divide the polygon interior into a certain number of closed sub regions. Because there is no L-shape and each of the polygon vertex must have at least one cut line, all sub regions have only 90-degree corners in the interior of polygon. Meanwhile, the non-inflection vertices of the polygon are 90-degree as well. Hence all the closed sub regions that compose of the partition of the given polygon are rectangles. which forms a rectangular partition on HGP. Because the PT edges on the polygon boundaries have zero cost, the PT and the resulting partition have the same cost.

($\Longleftarrow$) **An acyclic partition on HGP corresponds to a PT with same cost.** Let G_1 be the graph that the cut lines of the partition forms on HGP and R_1 ($R_1 \subseteq R$) be the set of the R-vertices in G_1. Firstly, G_1 must includes all the I-vertices of the polygon. Otherwise, the cut lines forming G_1 would not be a rectangular partition. Secondly, all the points in $R - R_1$ are isolated points from the graph G_1, each of which can be connected to G_1 by adding portion of polygon boundaries between

it and its closest I-vertex. Thus a new graph G_2 is formed. Because each point in $R - R_1$ is connected to only one vertex in graph G_1, and there is no additional connections between any two points of $R - R_1$, the newly formed graph G_2 has no cycle as well. Finally, if G_2 is a connected graph, it is a tree covering $I \cup R$; otherwise it is a forest which is formed by a set of subtrees. Since the partition is anchored partition, the subtrees are all anchored on polygon boundary. So, any two subtrees can be connected via polygon boundaries to form a tree G_3 covering $I \cup R$. Since edges on polygon boundary have 0-cost and do not introduce any L-shapes in interior, the final tree is on HGP, has no L-shape, covers $I \cup R$, and has the same cost with the partition. By definition, it is a PT. □

Theorem III.1 *MPT is equivalent to minimum cut length acyclic partition.*

Proof. ($\Longrightarrow$) **Given an MPT, it must correspond to an minimum acyclic partition with the same cost.** From Lemma III.1, the MPT with cost t_m must correspond to an acyclic partition on HGP with cost p_m and $p_m = t_m$. If the partition with cost p_m is not the minimum acyclic partition, there must be an acyclic partition with $p_i < p_m$. From the Lemma III.1, p_i must correspond to a partition tree with cost $p_i = t_i$. Hence $t_i < t_m$. This contradicts to the fact that the PT with t_m is an MPT. Therefore the corresponding partition is the minimum acyclic partition.

($\Longleftarrow$) **Given an minimum acyclic partition, it must correspond to an MPT with the same cost.** From Lemma III.1, the given minimum acyclic partition with cost p_i must correspond to a PT with cost t_i, and $p_i = t_i$. If the PT with cost t_i is not the minimum, there must be an MPT with cost $t_m < t_i$. From Lemma III.1, MPT with t_m corresponds to an acyclic partition with cost p_m, and $p_m = t_m$. Therefore, we have $p_m < p_i$. This contradicts to the fact that the partition with cost p_i is the minimum acyclic partition. Therefore, the PT is an MPT. □

Theorem III.1 indicates that we can solve the MCLRP problem by solving the MPT problem if the polygon has an MCLRP being an acyclic partition. In the case when there is no MCLRP being an acyclic partition, the problem can be solved suboptimally either with the MPT suggested acyclic minimal partition, or by post processing the constructed MPT to form cycles.

Note that the Steiner minimal tree problem is a NP-complete problem in general case [12]. And the MCLRP is a NP-hard problem in general case [10]. Therefore the computational complexity of solving MPT in practice is a big issue. In the next section, we will give an analysis on how to reduce the size of search space for solving the MPT problem.

IV. Reducing The Solution Search Space

For any optimization problem, the computational complexity and the quality of results of a heuristic algorithm are very much affected by the properties of the solution search space for the optimal solution. Two extremely important properties of the search space are the number of local optima and the "depth" of local optima [13]. They indicate the scale and the nature (smoothness or lack of) of the search space. In order to avoid the search algorithm being stuck at a local optimum one efficient way is to reduce the size of the search space. This will not only reduce the number and "depth" of the local optima, but it will also reduce the computational complexity of the algorithms searching for the optimal solution since it reduces the number of candidate solutions.

The size of the MPT search space is directly related to the number of the terminals connected by the tree. From the formulation, this variant Steiner tree uses the set $I \cup R$ as tree terminals. By definition, each I-vertex has two corresponding R-vertices formed by horizontal and vertical cut rays from it. If there are N I-vertices in the polygon, the problem size will be $3N$ for MPT. Reducing the problem size can effectively improve the efficiencies of MPT solvers.

The size of the search space is also related to the number of candidate solutions. By analyzing the MPT properties we can reduce the search space by removing the redundant candidate PTs from the search space. As long as the optimal solution is still in the reduced subset of the search space, this effort of reducing the search space will not affect the quality of results but will only improve the computational efficiency.

Below are two of the very useful and special properties associated with the formulated MPT. These properties can help us reduce the size of the MPT problem and the size of candidate solution set. Due to page limit, we omit the proofs that the reduced solution search space according to these two properties still contain the optimal solution MPT.

1. For any two partition trees, if the trees have the same cut-state portions in spite of different boundary-overlapping portions, we only need to keep one of them in the search space.
2. In an MPT, if there is a cut-state edge e having one endpoint as an R-vertex, the cut line that contains e must have the other endpoint to be that R-vertex's parent I-vertex.

The first property indicates that the solution search space for MPT can be reduced by keeping only one of the several PTs as candidate if these several PTs have same portions of tree edges in the polygon interior. The second property implies that the problem size of MPT can be reduced by not considering the full connections between all the terminals in the set $I \cup R$ during tree construction. Each R-vertex can only be directly connected to its parent I-vertex rather than to all the other I-vertices and R-vertices. Therefore the problem size will be far less than $3N$.

In addition to the above two properties, the search space can be reduced further by using the properties in Theorem II.1. Since MPT is equivalent to an acyclic type of MCLRP it must have the same properties as stated in Theorem II.1. Therefore any PT that has two cut-lines on one inflection vertex (except

when it has two chords) is not an MPT, and thus can be removed from the candidate PT set for MPT.

V. A Generic MPT Solver and A Linear-Time Heuristic for MPT

The generic MPT algorithm contains two core parts. The first part is the sub-tree fetcher and the second part is the tree cost evaluator. The sub-tree fetcher is used to get a sub Steiner tree rooted by a given tree node. The evaluator uses the cost function which considers all the constrains illustrated in the generic optimization formulation of MPT in Section IV to give the cost of the current sub-tree. The sub-tree fetcher determines the way the MPT algorithm constructs the final tree, and it can be used to construct the tree edge by edge, or sub-tree by sub-tree. Thus many rectilinear Steiner minimal tree construction algorithms can be applied here as long as the algorithm's cost evaluator is adapted to the sub-tree fetcher.

In this paper we adopt an L-path based suboptimal rectilinear Steiner tree construction algorithm [2] as a example to demonstrate the generic MPT algorithm flow. The formulation of the edge cost is modified according to the constraints in the generic optimization formulation of the MPT where the partition quality constraints required by MDP should be also added. The efforts listed in Section IV for reducing the scale of the problem and the search space are utilized in our heuristics to improve the efficiency of the algorithm.

In [2], it is proved that the computational complexity of the rectilinear Steiner tree construction is $O(n)$ on a separable minimal spanning tree with n nodes. We use an $O(n \log n)$ nearest neighbor algorithm [14] to build a separable minimal spanning tree to determine the connectivity between the I-vertices. Then we construct the MPT on this separable minimal spanning tree after adding R points to their parent I-vertices in spanning tree. We also prove that the computational complexity of MPT construction from the separable minimal spanning tree is $O((3N))$ where N is number of I-vertices. Due to length restriction, the proof is omitted here.

The following flow summarizes the proposed algorithm.

1. Read in the polygon, collect inflection vertices, store polygon edges, and generate the R vertices. One inflection vertex corresponds to only two R vertices.
2. Use all the I-vertices vertices as terminals. Build separable minimal spanning tree.
3. Connect all the R-vertices to the tree with their respective parent I-vertices only.
4. Starting from a leaf vertex of the separable minimal spanning tree do the following recursively:
   - For each vertex, enumerate all possible polygon-interior rectilinear path combinations of this vertex and all its children vertices.
   - Choose the combination with smallest total cost.
   - The cost is determined by the definition III.1, and all the partition quality constraints should be addressed as well.
   - The constraints in Section IV are enforced to eliminate some unnecessary combinations and to speed up the execution of the algorithm.

VI. Experimental Results

We ran the L-path based heuristic on a Redhat Linux server with two 2.8GHZ AMD Opteron CPUs and 8GB memory. We tested our algorithm on many hand-crafted as well as on industry examples. Many of them were large and complex. We also used other MDP tools which use traditional polygon fracturing methods on these same examples. The comparison shows that our proposed new algorithm has similar run time but has better quality of results than the that of other tools used in the form of fewer slivers, better fracturing uniformity, and smaller aspect ratio of the final figures, etc. Due to confidentiality issues, we cannot show the results obtained from the standard industry tools used. In this section we will only show the results of our own algorithm. We demonstrate 5 representative benchmarks for the examples that we have tested. Examples p3 and p5 are created manually; the other 3 examples are from real industry circuits.

The benchmark characteristics are shown in Table I, including the number of polygons, the number of holes, the number of original polygon vertices, and the number of polygon inflection vertices. Of those, the number of inflection vertices is an indicator of the problem size. The threshold for sliver size is also listed in Table I. We define the sliver threshold (unit is nm) as the minimum allowable size of a printable feature specified by the mask shop. If the smallest dimension of any partitioning figure is smaller than this value, it is considered to be a sliver.

TABLE I
The benchmark characteristics.

Bench	#poly	#hole	#vertex	#I-vertex	sliver threshold
p3	1	1	18	9	1
p5	1	0	26	11	1
ex02	1	1470	18260	12068	100
ex05	7	0	792	382	100
ex12	471	0	19524	9070	100

Using the listed sliver thresholds, Table II lists the quality of results generated by our MPT algorithm. The 2nd column is the number of partitioned rectangles, which is also referred to as the final figure count. The 3rd column is the total number of slivers. Among the slivers, there are two types: embedded slivers and edge slivers. The embedded slivers are the slivers that are in between two big figures. The edge slivers are the ones on the side of polygon. The embedded slivers can be handled by the

mask writing tools. They will not cause quality issues. The real problematic ones are the edge slivers. In column 4 of Table II, we listed the number of edge slivers as a criteria to indicate the partitioning quality.

TABLE II
THE PARTITION RESULTS.

Bench	#rectangle	#sliver	#edge sliver	# CPU(s)
p3	6	2	0	0
p5	12	2	2	0
ex02	7568	35	35	3
ex05	343	83	6	0.2
ex12	8792	19	1	0.5

Figure 1 shows the full layout of the partition results on the two hand-crafted examples. The solid lines are the original polygon edges. The dashed lines are the partition cut lines. Example "p3" (Figure (a))demonstrates how the polygon with holes are handled during partition. Example "p5" (Figure (b)) demonstrates partition of a more complex example and consideration of sliver constraints. Note that there often exist some unavoidable slivers in practical benchmarks. The preferences among all unavoidable slivers is as follows. An embedded sliver is better than an edge sliver; an edge sliver with a shorter total exposed boundary length is better. These preferences are incorporated in our algorithm.

The edge slivers in ex02 are all unavoidable, which is related to the benchmark itself. For other two realistic benchmarks ex05 and ex12, the edge sliver number is very small. Due to confidentiality reason, we do not show the snapshots of the partition results from these 3 industry examples (ex02, ex05, and ex12). Their result statistics can be found in Table II.

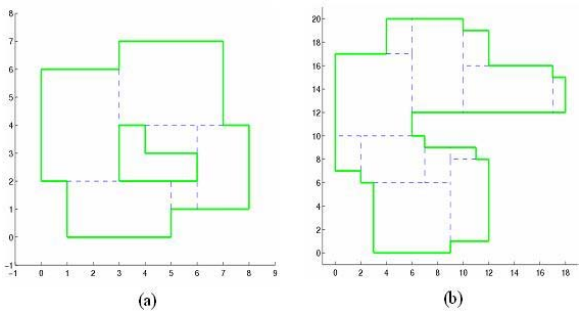

Fig. 1. Partition results on two simple benchmarks. Figure (a) is for "p3" and (b) is for "p5".

VII. CONCLUSIONS

This paper presents an efficient Steiner tree based approach for rectangular polygon partitioning. We solve the traditional polygon partitioning problem as a variant Steiner minimal tree: minimal partition tree (MPT). The equivalence between the MPT and the minimum cut length rectangular partition is proved. The properties of MPT are analyzed and utilized to reduce the size of the corresponding search space. Based on theoretical and practical analysis a generic algorithm and a linear time heuristic algorithm are proposed to construct the MPT. The experimental results clearly demonstrate the very high quality of the partitioned results. In future work we plan to extend this algorithm to handle non-rectilinear polygons as well.

REFERENCES

[1] A. B. Kahng and G. Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design*, 11:893–902, 1992.

[2] J. Ho, G. Vijayan, and C. K. Wong. New algorithms for the rectilinear Steiner tree problem. *IEEE Transactions on Computer-Aided Design*, 9:185–193, 1990.

[3] Hai Zhou. Efficient Steiner tree construction based on spanning graphs. *IEEE Transactions on Computer-Aided Design*, 23(5):704–710, 2004.

[4] Chris C. N. Chu. FLUTE: Fast lookup table based wirelength estimation technique. In *International Conference on Computer-Aided Design*, pages 696–701, 2004.

[5] T. Asano and H. Imai. Partitioning a polygonal region into trapezoids. *Journal of ACM*, 33:290–312, 1986.

[6] T. Ohtsuki. Minimum dissection of rectilinear regions. In *Proc. ISCS*, pages 1210–1213, 1982.

[7] J. O'Rourke, I. Pashchenko, and G. Tewari. Partitioning orthogonal polygons into fat rectangles. In *Proc. of 13th Canadian Conf. on Comp. Geom.*, pages 133–136, 2001.

[8] P. D. Buck M. Bloecker, R. Gladhill and etc. Metrics to assess fracture quality for variable shaped beam lithography. In *proc. SPIE international society of optical engineering*, volume 6349, 2006.

[9] A. Lingas, R. Pinter, R. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proc. 20th Allerton Conf. on Communication, Control, and Computing*, pages 53–63, 1982.

[10] Ding-Zhu Du and Ker-I Ko. Chapter 7: Adaptive partition. In *Design and Analysis of Approximation Algorithms*, 2002.

[11] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14:255–265, 1966.

[12] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

[13] Gelsey A. and Smith D. A search space toolkit. In *Proceeding of 11th Conference on Artificial Intelligence for Applications*, pages 117–123, 1995.

[14] L.J. Guibas and J. Stolfi. On computing all northeast nearest neighbors in the l1 metric. *Information Processing Letters*, 17:219–223, 1983.