EFFICIENT APPROXIMATION ALGORITHMS FOR FLOORPLAN AREA MINIMIZATION

Danny Z. Chen[*] *Xiaobo (Sharon) Hu*[†]

ABSTRACT

Approximation has been shown to be an effective method for reducing the time and space costs of solving various floorplan area minimization problems. In this paper, we present several approximation techniques for solving floorplan area minimization problems. These new techniques enable us to reduce both the time and space complexities of the previously best known approximation algorithms by more than a factor of n and n^2 for rectangular and L-shaped sub-floorplans, respectively (where n is the number of given implementations). The efficiency in the time and space complexities is critical to the applicability of such approximation techniques in floorplan area minimization algorithms. We also give a technique for enhancing the quality of approximation results.

1. INTRODUCTION

Floorplan area minimization plays an important role in achieving an optimal floorplan for an integrated circuit. Most floorplan area minimization algorithms search for the minimum area implementation by first constructing a list of *all* non-redundant implementations of the entire floorplan (e.g., [4, 5, 6, 7, 9]). The list is built recursively in a bottom-up fashion, by traversing the tree (called *floorplan tree*) that represents the floorplan under consideration. Depending on the number of implementations for the basic circuit modules and the structural complexity of a floorplan, the size of such a list can become very large [6, 10]. Both the computation time and memory usage of the known floorplan area minimization algorithms are directly proportional to the size of the list consisting of the non-redundant implementations. For large floorplans, such algorithms may fail to produce a solution due to exorbitant space or time cost.

An effective method for improving the efficiency of the known floorplan area minimization algorithms and, more importantly, for enabling such algorithms to handle large floorplans, is to *approximate* the list of all non-redundant implementations associated to an internal node of a given floorplan tree by a new list that contains a smaller number of non-redundant implementations. Wang and Wong [10] have proposed several approximation algorithms for two important types of sub-floorplans: one in which the list (called *R-list*) consists of rectangular implementations and the other in which the list (called *L-list*) consists of L-shaped implementations. They used an interesting graph-theoretic approach. Let n be the number of non-redundant implementations in the original list and k the number of implementations in the resulting list after approximation. Wang and Wong's algorithms require $O(kn^2)$ time and $O(n^2)$ space for approximating an R-list, and $O(n^3)$ time and $O(n^2)$ space for approximating an L-list.

In this paper, we propose new algorithmic techniques for solving the approximation problems for both the R-lists and L-lists. In contrast to Wang and Wong's graph-theoretic approach [10], our approach hinges more heavily on geometric structures of these approximation problems. Specifically, we prove a number of observations on useful geometric structures of the approximation problems for the R-lists and L-lists. By exploiting these geometric properties, we develop approximation algorithms that are substantially more efficient in both the time and space complexities than the previously best known results [10]. Regardless of the type of the lists, our following results hold true. A simple-to-implement version of our algorithms has a time complexity of $O(nk)$ and a space complexity of $O(n)$. A more sophisticated version of our algorithms runs in $n2^{O(\sqrt{\log k \log\log n})}$ time (which is equivalent to $O(nk^{\epsilon})$ for any constant $\epsilon > 0$) and still takes $O(n)$ space. Our improvements on the time and space complexities of the approximation algorithms for the R-lists and L-lists make it more applicable and attractive to incorporate the approximation techniques into floorplan area minimization algorithms. Furthermore, these results are obtained without sacrificing the quality of the approximation solutions. Actually, our algorithms generate better approximation results than those of [10] (i.e., our results have smaller approximation errors) when applied to the set of irreducible L-lists of an L-shaped block.

2. PROBLEM FORMULATION

We first review some needed terminology and useful structures (many of which are from [10]), and then formulate the floorplan approximation problems studied in this paper. The basic terminology related to floorplanning can be found in many publications (e.g., [9]).

We consider two commonly-used sub-floorplans: *rectangular* and *L-shaped* blocks [9]. An implementation (i.e. a layout alternative) of a rectangular block can be represented by a 2-tuple (w, h), where w is its width and h is its height. An implementation of an L-shaped block can be represented by a 4-tuple (w^1, w^2, h^1, h^2), where w^1 and w^2 are the widths of the two horizontal edges, and h^1 and h^2 are the heights of the two vertical edges, with $w^1 \geq w^2$ and $h^1 \geq h^2$.

Wang and Wong [10] used an *R-list* to represent multiple implementations of a rectangular block. An implementation list $\{(w_1, h_1), (w_2, h_2), \ldots, (w_n, h_n)\}$ is an *R-list* if $w_i \geq w_j$ and $h_i \leq h_j$ for all $1 \leq i < j \leq n$. Similarly, multiple implementations of an L-shaped block can be represented by an *L-list* [10]. An implementation list $\{(w_1^1, w_1^2, h_1^1, h_1^2), (w_2^1, w_2^2, h_2^1, h_2^2), \ldots, (w_n^1, w_n^2, h_n^1, h_n^2)\}$ is an *L-list* if $w_i^1 \geq w_j^1$, $w_i^2 = w_j^2$, $h_i^1 \leq h_j^1$, and $h_i^2 \leq h_j^2$ for all $1 \leq i < j \leq n$. Note that the implementations of an L-shaped block may be represented by multiple L-lists.

An R-list is called an *irreducible* R-list [10] if there are no redundant implementations in the list. An implementation (w, h) is *redundant* if there exists another implementation (w', h') such that $w \geq w'$ and $h \geq h'$. An *irreducible* L-list can be defined similarly [10]. Irreducible R-lists and irreducible L-lists are fundamental data structures that are used by a number of floorplan area minimization algorithms to store non-redundant implementations that may lead to optimal floorplan implementations. For large, complex floorplans, the number of non-redundant implementations in such lists can be enormous [6, 9].

To reduce the number of non-redundant implementations or the size of an irreducible R-list or L-list, an approxima-

[*]Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, U.S.A.

[†]Department of Electrical and Computer Engineering, Western Michigan University, Kalamazoo, MI 49008, U.S.A.

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

tion approach was proposed by Wang and Wong [10]. A predefined limit is set for the number of non-redundant implementations that can be stored at an internal node of a floorplan tree (this limit may be determined based on the available time and space resources). When the number of non-redundant implementations at the internal node exceeds this limit, only a subset of these non-redundant implementations will be retained for the subsequent steps while the others are eliminated. Clearly, the retained non-redundant implementations can only approximate the entire non-redundant implementation set. Hence, an important problem is to select a subset of implementations that best approximates the original set of implementations. This problem is formulated formally as follows.

Let $R = \{r_1, r_2, \ldots, r_n\}$ be an irreducible R-list, where $r_i = (w_i, h_i)$, and $L = \{l_1, l_2, \ldots, l_n\}$ be an irreducible L-list, where $l_i = (w_i^1, w_i^2, h_i^1, h_i^2)$, for $i = 1, 2, \ldots, n$. Given an integer k, $k < n$, we denote a subset of k implementations of R (resp., L) by $\tilde{\mathrm{R}} = \{r_{d_1}, r_{d_2}, \ldots, r_{d_k}\}$ (resp., $\tilde{\mathrm{L}} = \{l_{d_1}, l_{d_2}, \ldots, l_{d_k}\}$), where $1 = d_1 < d_2 < \cdots < d_k = n$. Without cluttering the exposition, our following discussion will use R-lists only. If L-lists are concerned, one only needs to replace R with L and $\tilde{\mathrm{R}}$ with $\tilde{\mathrm{L}}$ in our exposition. Clearly, the list $\tilde{\mathrm{R}}$ is an approximation to the original R-list R. More precisely, an error is committed when implementations $r_{d_z} = r_i$ and $r_{d_{z+1}} = r_j$ but not those implementations of R strictly between r_i and r_j are included in $\tilde{\mathrm{R}}$, for each $z = 1, 2, \ldots, k-1$. Let the error induced by including in $\tilde{\mathrm{R}}$ the implementations $r_{d_z} = r_i$ and $r_{d_{z+1}} = r_j$ but not the implementations strictly between r_i and r_j be denoted by $e(d_z, d_{z+1}) = e(i, j)$. Then the total approximation error committed by using $\tilde{\mathrm{R}}$ to replace R is simply

$$E(R, \tilde{\mathrm{R}}) = \sum_{z=1}^{k-1} e(d_z, d_{z+1}) \qquad (1)$$

Therefore, the problem that we are concerned with is to obtain a list $\tilde{\mathrm{R}}$ that results in the minimum approximation error $E(R, \tilde{\mathrm{R}})$. It should be pointed out that the actual value of each $e(i, j)$ depends on the specific error definition and will be discussed in Section 4.

3. MAIN IDEAS AND TECHNIQUES

Several new ideas are used in our approach to finding the best approximate sub-list of size k for a given R-list (resp., L-list) of size n, $k < n$. One of them is to reduce this approximation problem to that of computing the minimum-weight K-link path in a complete, weighted, directed acyclic graph (DAG) possessing certain special geometric properties [2, 8]. Another key idea is to develop a scheme for implicitly representing the complete, weighted DAG. This scheme stores the graph of $O(n^2)$ weighted edges in only $O(n)$ space, yet still allows graph information to be extracted as if an explicit graph representation is available.

Considering the problem of approximating an irreducible R-list R of size n, we model it by a complete, weighted DAG G as follows: Each vertex v_i in G corresponds to the implementation r_i in the R-list R and each arc $a(i, j)$ in G has a weight equal to the approximation error $e(i, j)$ (for all $1 \leq i < j \leq n$). It is not difficult to see that, for an integer k, $0 < k < n$, the vertices along $P_{k-1}(1, n)$, the minimum-weight $(k-1)$-link path from v_1 to v_n in G, correspond precisely to those k implementations that constitute the best approximation $\tilde{\mathrm{R}}$ to the original R-list R. (A path $P_K(i, j)$ in G is a minimum-weight K-link path from v_i to v_j if $P_K(i, j)$ consists of exactly K arcs and has the minimum total weight among all K-arc paths from v_i to v_j.)

Although the DAG model we have just described is similar to the one used in [10], we are able to prove a number of useful geometric structures about the R-list (resp., L-list) approximation problem. Our first observation is that the complete, weighted DAG G that models an R-list or L-list has an important geometric structure called the Monge property [2, 8]. (We shall present the details in the next section.) Let a complete, weighted DAG G have vertices v_1, v_2, ..., v_n and a weight $w(i, j)$ for each arc $a(i, j)$, $i < j$. Then, G is said to satisfy the *concave Monge property* if for all $1 < i + 1 < j < n$, the following inequality holds [2, 8]:

$$w(i, j) + w(i+1, j+1) \leq w(i, j+1) + w(i+1, j). \qquad (2)$$

Aggarwal, Schieber, and Tokuyama [2] and Schieber [8] have shown that the minimum-weight K-link path problem on a complete, weighted DAG with the concave Monge property can be solved by faster algorithms than those for a general graph [3]. *Provided that the graph is already available,* the best known algorithm for this problem takes $n2^{O(\sqrt{\log k \log\log n})}$ time, which is bounded by $O(nk^{\epsilon})$ for any constant $\epsilon > 0$, and $O(n)$ space [8].

It should be pointed out that although the complete, weighted DAG G that models an R-list (resp., L-list) can be shown to have the concave Monge property, simply applying the algorithms in [2, 8] to G would not achieve the improvement on the time and space complexities that we claim. In fact, such a solution would take $O(n^2)$ time and space (resp., $O(n^3)$ time and $O(n^2)$ space) for the R-list (resp., L-list) approximation. The reason is that one would have to make the graph G available first, and explicitly constructing G, as shown in [10], takes $O(n^2)$ time and space (resp., $O(n^3)$ time and $O(n^2)$ space) for an R-list (resp., L-list). Our idea for overcoming this difficulty is, instead of building the graph explicitly (and hence paying the high price for the graph construction and representation), representing the graph *implicitly*. More specifically, by taking advantage of several additional geometric properties of the approximation problems, we design a scheme for implicitly representing the graph G. Our implicit representation of G requires only $O(n)$ time and space to construct, yet it enables us to extract information of G as if an explicit representation of G is available. Combining our observations on the Monge property of the graph we use with our implicit graph representation schemes, we solve the R-list and L-list approximation problems in the claimed $n2^{O(\sqrt{\log k \log\log n})}$ time (which is bounded by $O(nk^{\epsilon})$ for any constant $\epsilon > 0$) and $O(n)$ space.

Note that Schieber's algorithm [8] for computing the minimum-weight K-link path in a complete, weighted DAG with the concave Monge property makes use of a variant of the parametric search technique which is quite challenging for practical implementation. To resolve this issue, we present a different version of the algorithms for the R-list and L-list approximation problems that are simple to implement yet reasonably efficient. These simple-to-implement algorithms are also based on the concave Monge property of the graph we use and on our implicit graph representation schemes. However, we replace in these algorithms the parametric search technique [8] by the fast Monge matrix searching algorithm [1] and a novel path construction scheme based on a divide and conquer strategy. In consequence, these simple-to-implement algorithms require $O(nk)$ time and $O(n)$ space, slightly worse than the time complexity of our sophisticated algorithms but still significantly better than the previously best known results [10]. The ideas and details of the simple-to-implement algorithms are interesting in their own right but due to the space limitation, we

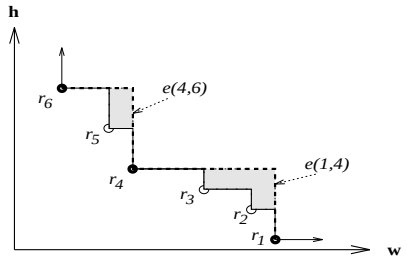

Figure 1. Staircase curves for R-list $R = \{r_1, r_2, r_3, r_4, r_5, r_6\}$ and $\tilde{R} = \{r_1, r_4, r_6\}$ and the approximation errors.

will omit the detailed discussion of these algorithms.

4. FLOORPLAN APPROXIMATION ALGORITHMS

In this section, we substantiate the main ideas and techniques outlined in the previous section. We will discuss the R-list approximation problem first but omit the proofs to our lemmas and theorems. Due to the page limit, we will only briefly present our approach to solving the more complicated L-list approximation problem.

4.1. R-list approximation

We begin with defining how the actual values of approximation errors are determined. We use the same error definition as introduced by Wang and Wong [10]. Quantitatively, the approximation error associated with two consecutive implementations r_{d_i} and $r_{d_{i+1}}$ in $\tilde{R}$ due to the omission of those implementations in R strictly between r_{d_i} and $r_{d_{i+1}}$ is defined as follows. If $d_i + 1 < d_{i+1}$, then

$$e(d_i, d_{i+1}) = \sum_{q=d_i+1}^{d_{i+1}-1} (w_{d_i} - w_q)(h_{q+1} - h_q) \qquad (3)$$

and $e(d_i, d_{i+1}) = 0$ when $d_i + 1 = d_{i+1}$. Then, the total error of approximating R with $\tilde{R}$ is

$$E(R, \tilde{R}) = \sum_{i=1}^{k-1} e(d_i, d_{i+1}). \qquad (4)$$

Graphically, the original R-list R and an approximating R-list $\tilde{R}$ can be represented by two *staircase curves* (see Figure 1). The total error committed by approximating R with $\tilde{R}$ is indicated by the shaded regions in Figure 1, representing the area enclosed by the two staircase curves.

Although the above definition of $e(d_i, d_{i+1})$ implies a straightforward way of computing $e(d_i, d_{i+1})$, we give a lemma which shows a different way of calculating $e(d_i, d_{i+1})$. This lemma is also critical to our presentation of the geometric properties of the R-list approximation problem. It is based on the observation that the error between two consecutive implementations r_{d_i} and $r_{d_{i+1}}$ in $\tilde{R}$ is directly related to the area difference between the rectangle with width w_{d_i} and height $(h_{d_{i+1}} - h_{d_i})$ and the rectilinear polygon under the staircase curve for R between r_{d_i} and $r_{d_{i+1}}$ (see Figure 1).

Lemma 1 *For an R-list R, $R = \{r_1, r_2, \cdots, r_n\}$ and each $r_j = (w_j, h_j)$, let $a_i = w_i(h_{i+1} - h_i)$ for $i = 1, 2, \ldots, n-1$. The approximation error $e(i, j)$ (due to including r_i and r_j in $\tilde{R}$ and omitting all the implementations in R strictly between the two), $i < j$, can be calculated as follows:*

$$e(i, j) = w_i(h_j - h_i) - \sum_{q=i}^{j-1} a_q$$

Based on Lemma 1, we can show that the complete, weighted DAG G we use to model the R-list approximation problem satisfies the concave Monge property. This is stated in Lemma 2.

Lemma 2 *Given an R-list $R = \{r_1, r_2, \cdots, r_n\}$, let G be the complete, weighted DAG that models the R-list approximation problem on R. Then, G satisfies the concave Monge condition. That is, for all $1 < i + 1 < j < n$, the following inequality holds:*

$$e(i, j) + e(i+1, j+1) \leq e(i, j+1) + e(i+1, j)$$

where $e(i, j)$ is as defined in Lemma 1.

As pointed out in the previous section, having the Monge property alone would not result in solving the R-list approximation problem within the claimed time and space complexities. We need to develop a scheme for implicitly representing the complete, weighted DAG we use. In the implicit graph representation scheme, a preprocessing step is used to calculate and store a value $A_i = \sum_{q=1}^{i-1} a_q$ (where a_q is as defined in Lemma 1) for each $i = 2, 3, \ldots, n$ (with $A_1 = 0$). When the error associated with a pair of implementations r_i and r_j is needed, it can be computed in constant time by using

$$e(i, j) = w_i(h_j - h_i) - (A_j - A_i). \qquad (5)$$

The correctness of Equation (5) follows immediately from Lemma 1. The preprocessing step takes only linear time and space for computing the A_i's. Therefore, we reduce the time and space for constructing and storing the complete, weighted DAG G from $O(n^2)$ to $O(n)$. (In fact, this implicit graph representation scheme can be applied to any R-list algorithms that use the same error definition as (3).)

We summarize in Figure 2 our R-list approximation algorithm, where `Find_Min_K_Path`(G, K) denotes the minimum-weight K-link path algorithm given in [8]. Note that the complete, weighted DAG G is represented implicitly as discussed above, that is, only linear space is needed for storing G. When the weight of an arc in G is required by `Find_Min_K_Path`, (5) is evaluated in constant time. The time and space complexities of the algorithm is summarized in Theorem 1.

Algorithm: `R-Approximation`
Input: An irreducible R-list $R = \{r_1, r_2, \cdots, r_n\}$, where each $r_i = (w_i, h_i)$, and a positive integer $k < n$;
Output: An R-list $\tilde{R}$ containing k implementations from R, which minimizes the total approximation error $E(R, \tilde{R})$;
begin
  $A(1) \leftarrow 0$;
  for $i = 2$ to n
    $A(i) \leftarrow A(i-1) + w_{i-1}(h_i - h_{i-1})$;
  $G \leftarrow$ the complete, weighted DAG for R;
  $P \leftarrow$ `Find_Min_K_Path`$(G, k-1)$;
  output the k vertices on the path P;
end

Figure 2. Algorithm for finding the best approximation to a given R-list R of size n and an integer k $(k < n)$.

Theorem 1 *Given an R-list R of size n and an integer k ($k < n$), in $n2^{O(\sqrt{\log k \log\log n})}$ time and $O(n)$ space, the algorithm in Figure 2 selects k elements from R to form an R-list $\tilde{R}$ of size k which best approximates R.*

4.2. L-list approximation

Although the L-list approximation problem in principle can be solved in a similar fashion as the R-list approximation, the details are substantially different. The major difficulty lies in the more complex computation required by the definition of approximation error in the L-list case. It not only introduces new challenges in demonstrating that the L-list approximation problem has the required properties but also complicates the implicit graph representation scheme.

By using the same error definition for an L-list L as in [10], the approximation error associated with two consecutive implementations l_{d_i} and $l_{d_{i+1}}$ in $\tilde{\mathrm{L}}$ by discarding those implementations in L strictly between l_{d_i} and $l_{d_{i+1}}$ of L is determined by

$$e(d_i, d_{i+1}) = \sum_{q=d_i+1}^{d_{i+1}-1} cost(l_q) \tag{6}$$

where

$$cost(l_q) = \min\left\{dist(l_{d_i}, l_q), dist(l_q, l_{d_{i+1}})\right\} \tag{7}$$

$$dist(l_i, l_j) = |w_i^1 - w_j^1| + |h_i^1 - h_j^1| + |h_i^2 - h_j^2| \tag{8}$$

The total error of approximating L with $\tilde{\mathrm{L}}$ is hence

$$E(L, \tilde{\mathrm{L}}) = \sum_{i=1}^{k-1} e(d_i, d_{i+1}). \tag{9}$$

The following important observation forms the basis for our approach to solving the L-list approximation problem,

Lemma 3 *Let $(l_i, l_{i+1}, \ldots, l_j)$ be a subsequence of implementations in an irreducible L-list L. There exists an index s, $i \le s \le j$, such that $dist(l_i, l_p) \le dist(l_p, l_j)$ for any p, $i \le p \le s$ and that $dist(l_i, l_q) > dist(l_q, l_j)$ for any q, $s < q \le j$. Such an index s is called a* separating point.

Based on 3, we are able to show that the complete, weighted DAG G for modeling the L-list approximation problem has the concave Monge property (similar to Lemma 2.)

To apply the techniques outlined in Section 3 to the L-list approximation problem, we also need to have an implicit graph representation scheme which requires only $O(n)$ space to store G. (In [10], $O(n^3)$ time and $O(n^2)$ space are used to build and store the associated DAG.) The following lemmas make it possible for us to achieve this goal. We skip the details of our representation scheme and the actual algorithm due to the space limitation.

Lemma 4 *For an irreducible L-list $L = \{l_1, l_2, \ldots, l_n\}$, let*

$$X(i) = \sum_{j=1}^{i} dist(l_1, l_j),\ i = 1, 2, \ldots, n$$

$$Y(i) = \sum_{j=i}^{n} dist(l_j, l_n),\ i = n, n-1, \ldots, 1$$

We then have, for any i and j with $i < j$,

$$\sum_{q=i}^{j} dist(l_i, l_q) = X(j) - X(i) - (j-i) \cdot dist(l_1, l_i)$$

$$\sum_{q=i}^{j} dist(l_q, l_j) = Y(i) - Y(j) - (j-i) \cdot dist(l_j, l_n)$$

Lemma 5 *Given the separating point s for two implementations l_i and l_j in an irreducible L-list with $i < j$, the approximation error $e(i, j)$ can be computed by*

$$e(i,j) = X(s) - X(i) - (s-i) \cdot dist(l_1, l_i) + Y(s+1) - Y(j) - (j-s-1) \cdot dist(l_j, l_n) \tag{10}$$

Lemma 6 *Given an irreducible L-list L, for implementations $l_i, l_q, l_j \in L$ and $i < q < j$, let s_{ij} be the separating point for l_i and l_j, s_{iq} be the separating point for l_i and l_q, and s_{qj} be the separating point for l_q and l_j. We have (1) $s_{qj} \ge s_{ij}$, and (2) $s_{iq} \le s_{ij}$.*

The time and space complexities of our L-list approximation algorithm are given in Theorem 2.

Theorem 2 *Given an L-list L of size n and an integer k ($k < n$), in $n2^{O(\sqrt{\log k \log\log n})}$ time and $O(n)$ space, our L-list approximation algorithm selects k elements from L to form an L-list $\tilde{L}$ which best approximates L.*

To apply the L-list approximation algorithm to the approximation problem in minimizing the floorplan area of an L-shaped block, note that the non-redundant implementations of an L-shaped block may need to be stored in a set of irreducible L-lists, rather than a single L-list [9]. Wang and Wong [10] proposed to allocate the number of implementations to be selected in proportion to the size of each L-list. The L-list approximation algorithm is then applied to each L-list. Although this approach is intuitive, it does not always produce implementations that best approximate the original implementations since the approximations of some L-lists may introduce larger errors than others.

Our approach is to construct a *single*, complete, weighted DAG G_L for *all* implementations represented by the multiple L-lists, rather than consider each L-list separately as in [10]. To solve the minimum-weight K-link path problem on G_L within the time and space complexities we claimed earlier, we need to verify on G_L that the concave Monge property still holds and that the implicit graph representation scheme is still applicable. The details for this will be left to the full paper.

REFERENCES

[1] A. Aggarwal, M.M. Klawe, S. Moran, P. Shor and R. Wilber, "Geometric applications of a matrix-searching algorithm," *Algorithmica*, vol. 2, 1987, pp. 195-208.

[2] A. Aggarwal, B. Schieber and T. Tokuyama, "Finding a minimum weight k-link path in graphs with Monge property and applications," *Proc. 9th Annual ACM Symp. on Computational Geometry*, 1993, pp. 189-197.

[3] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1979.

[4] T. Lengauer and R. Muller, "A robust framework for hierarchical floorplanning with integrated global wiring," *Proc. IEEE Int'l Conf. on Computer-Aided Design*, 1990, pp. 148-151.

[5] R.H.J.M. Otten, "Automatic floorplan design," *Proc. 19th ACM/IEEE Design Automation Conf.*, 1982, pp. 261-267.

[6] P. Pan and C.L. Liu, "Area minimization for floorplans," *IEEE Trans. on Computer-Aided Design*, vol. 14, no. 1, 1995, pp. 123-132.

[7] M. Pedram and B. Preas, "A hierarchical floorplanning approach," *Proc. IEEE Int'l Conf. on Computer Design*, 1990, pp. 332-338.

[8] B. Schieber, "Computing a minimum-weight k-link path in graphs with the concave Monge property," *Proc. 6th Annual ACM-SIAM Symp. on Disc. Algorithms*, 1995, pp. 405-411.

[9] T.-C. Wang and D.F. Wong, "Optimal floorplan area optimization," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 8, 1992, pp. 992-1002.

[10] T.-C. Wang and D.F. Wong, "Graph-based techniques to speedup floorplan area optimization," *Integration, the VLSI Journal*, vol. 15, 1993, pp. 179-199.