# The Reproducing Placement Problem with Applications

Wei-Liang Lin M. Sarrafzadeh C. K. Wong

Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

Watson Research Center
IBM Corp.
Yorktown Heights, NY 10598

## Abstract

*We study a new placement problem: the reproducing placement problem (RPP). In each phase a module (or gate) is decomposed into two (or more) simpler modules. The goal is find a "good" placement in each phase. The problem, being iterative in nature, requires an iterative algorithm. The problem finds applications in several gate-level placement problems, e.g., in layout-driven logic synthesis.*

*We introduce the notion of minimum floating Steiner trees (MFST). We employ an MFST algorithm as a central step in solving the RPP. A Hanan-like theorem is established for the MFST problem and two approximation algorithms are proposed. Experiments on commonly employed benchmarks verify the effectiveness of the proposed technique.*

## 1 Introduction

Placement is the problem of finding good locations for modules in a given circuit. Earlier approaches, ad-hoc in nature, served as a foundation for systematic approaches to the problem and resulted in coherent theories. Despite invaluable contributions a number of placement problems are not completely understood, especially variations dictated by present day technology. Here we study one such problem most suitable for gate-level placement.

The problem under investigation is called the *reproducing placement problem*, or, RPP. The input is a circuit consisting of an initial set of modules and a collection of multi-terminal nets. In each phase, in the placement process, a module may decompose into two (or more) smaller modules; we say the module is *reproduced*. As a result, the set of nets are also changing in each stage. Our goal is minimizing the *total length* of nets in each phase. As our emphasis is at the gate level, we shall use the terms modules and gates interchangeably. The reproducing placement problem serves as:

- **A guide in a layout-driven logic synthesis algorithm,**
- **A hierarchical stand-alone placement algorithm,**
- **A pre-processor or a post-processor for other placement algorithms.**

We will elaborate on one of the applications of RPP in later sections.

A number of effective placement algorithms have been proposed (see [SM91] for a survey). The RPP requires an iterative algorithm—a constructive method is time consuming and cannot utilize the placements obtained in previous phases. Existing iterative techniques are not suitable for our applications. They either view the problem locally or simplify the problem's objective (e.g., aim to minimize the total length of the bounding boxes of the nets instead of minimizing the actual lengths) to an extend where the quality of the result is sacrificed. Our experimental results verify that.

An instance of the *minimum floating Steiner tree* (MFST) problem consists of a set of gates and a collection of multi-terminal nets. The positions of all gates, called *fixed gates*, are known; except one gate, called *the floating gate*. The floating gate is incident to more than one net (typically, 3-8 nets in the layout-driven logic synthesis application, and 4-30 nets in the hierarchical placement application). The goal is to place the floating gate as to minimize the total wire length of all incident nets. We shall use the floating Steiner tree problem as a tool to solve an arbitrary instance of the RPP. An instance of the floating Steiner tree problem is shown in Figure 1a, where $gate(m)$ is the floating gate. Outputs of $gate(m)$ are $gate(O_1)$, $gate(O_2)$, $gate(O_3)$, its inputs are $gate(I_1)$ and $gate(I_2)$. There are three nets (one net shown

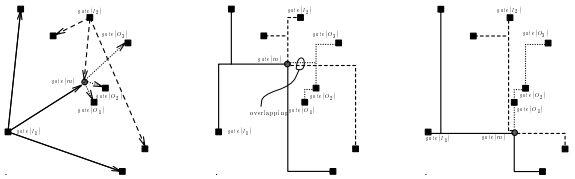


Figure 1: (a) An instance of the MFST problem. (b) A mass-center tree. (c) A floating Steiner tree.

with solid, one with dashed, and the last one with dotted lines). We have to place $gate(m)$ as to minimize the total (Steiner) length of the nets incident on the gate (i.e., total length of the three nets).

In the applications we are considering, gates are small enough that we view them as points (i.e., they occupy zero area). Here, all distances are measured in $L_1$ or rectilinear metric. Notice that the MFST problem is completely different from the one Steiner point problem [GP87, KR92], for in the MFST problem we deal with multiple nets simultaneously.

The next section formalizes the RPP, proposes an algorithm for it, and then, elaborates on one of its applications. Section 3 formalizes the MFST problem. A Hanan-like theorem is established. Two approximation algorithms with different performances and running times are proposed. Section 4 presents experimental results. The last section is devoted to conclusion.

## 2 An RPP Algorithm and an Application

We first formalize the reproducing placement problem and then propose an algorithm for solving it. Applications of the proposed technique to one problem will be discussed.

In Phase 0, we are given a circuit $C_0 = (G_0, N_0)$, consisting of a set $G_0$ of gates and a collection $N_0$ of multiterminal nets, where each net specifies a subset of terminals to be interconnected, that is, $N_0 \subseteq 2^{G_0}$. In Phase $i$, $C_{i-1}$ is transformed into $C_i$ by *reproduction* of one gate $g_j \in G_{i-1}$ into a collection of gates ($g_j$ is determined in this phase). The placements of $C_{i-1}$ and $C_i$ differ only in $g_j$ and the gates reproduced from $g_j$. There are a total of $\phi + 1$ phases. Our goal is to find a placement of $C_0, C_1, \ldots, C_\phi$ minimizing the total length of each circuit $C_i$, given the placement of $C_{i-1}$.

First, we place $C_0$ using a traditional placement algorithm. In our placement algorithm we solve each phase considering the previous phase and independent of future phases: in most of our application we do not know which gates will be reproduced. The algorithm we propose is summarized as follows. In Phase $i$, we consider gate $g_j$ that was just reproduced to form a collection of gates $g_j^1, \ldots, g_j^m$. We place gates $g_j^k$, $1 \leq k \leq m$, while maintaining the location of other gates (that were placed in previous phases). Placement of each gate $g_j^k$ is formulated as a MFST problem, to be discussed in the next section.

Next, we discuss one of the applications of the RPP. We should again emphasize that the following techniques are applicable to gate-level circuits (we have not studied their application to other classes of designs yet).

- **Layout-driven logic synthesis:** We consider logic synthesis and physical design simultaneously. Several algorithms have been proposed [LPP93, PB91, CCMS93] on this topic. The main idea is to simultaneously and iteratively "factor" the given Boolean function(s) and obtain an approximate placement. In each phase a function is partitioned into simpler functions, and gates realizing the simpler functions are properly placed. [PB91] employed a center-of-mass approach to place a gate: each gate is placed at the center-of-mass of all nets interconnecting it. Figure 1b shows a placement of the circuit in Figure 1a. The center-of-mass approach results in a placement with total length 1167, whereas the floating Steiner tree approach produces a placement with total length 1102, shown in Figure 1c. The RPP in connection with a logic factorization procedure can form an effective layout-driven logic synthesis paradigm. (See experimental results.)

Since the MFST problem is a fundamental aspect of the RPP, we shall elaborate on it in the next section.

## 3 Minimum Floating Steiner Tree Problem

In this section, the minimum floating Steiner tree (MFST) problem is formulated. A "multiple" Hanan grid is introduced. As a basis for understanding the MFST problem, we introduce the minimum floating spanning tree problem. Then two approximation algorithms are described.

**MFST Problem:**

**Input:** A collection of sets of fixed nodes $\{\{P1_1, P1_2, ..., P1_{i_1}\}, \{P2_1, P2_2, ..., P2_{i_2}\}, ..., \{Pm_1, Pm_2, ..., Pm_{i_m}\}\}$, where $m$ is the number of nets, and $i_j$ is the number of fixed nodes in net $j$. Fixed nodes within the same set belong to the same net.

**Output:** The position of a new node, floating node $P$, along with the layout of all nets.

The goal is to minimize the total wire length.

By passing a horizontal and a vertical line through each fixed node, we obtain a so-called *multiple Hanan grid*. Intersections of the lines define the grid points. Line segments connecting adjacent points are the grid edges.

The *minimum floating spanning tree* (MFSPT) problem is closely related to the MFST problem. The MFSPT problem differs the MFST problem by constructing each net as a minimum spanning tree instead of a Steiner tree and minimizing the total minimum spanning tree length instead of the total wire length.

Three theorems are introduced next.

**Theorem 1** *There exists an optimal solution with the position of the floating node at one of the multiple Hanan grid points.*

Theorem 1 implies that for our purpose we only need to consider a set of grid points. The next theorem shows the problem is still hard.

**Theorem 2** *The minimum floating Steiner tree problem is NP-hard.*

The following theorem provides a tool for solving the MFST problem.

**Theorem 3** *A minimum floating spanning tree is a $\frac{3}{2}$-approximation of the minimum floating Steiner tree problem.*

These theorems help to understand the following two algorithms.

A $\frac{5}{2}$-approximation algorithm is outlined as follows: First, we arbitrarily choose one fixed node from each net. A connection with minimum length is created based on the chosen nodes. Then, a minimum spanning tree is created for each net.

The other algorithm, constructing a MFSPT, is a $\frac{3}{2}$-approximation, as shown in Theorem 3. The basic idea is as follows: Considering each multiple grid point, we need to consider no more than 8 (fixed) neighbors of it. This limited number 8 is a consequence of Dirichlet partition [GP87, KR92]. An existing minimum spanning tree will be updated in constant time provided that $O(k^2)$ preprocession time is spent for each net, where $k$ is the number of fixed nodes for that particular net ($k \leq n$). Then, for each (floating) node under consideration we obtain a (total length) cost. There are $n^2$ multiple grid points, each needs $m$ updated minimum spanning trees, and the preprocession time is also $O(mn^2)$. Therefore, we obtain a $\frac{3}{2}$-approximation algorithm running in $O(mn^2)$ time.

## 4 Experimental Results

We have two types of experiments. The first considers placement of a single floating gate, and the second considers layout-driven logic synthesis.

In the first experiment, a collection of small circuits with 9 to 64 nodes are randomly generated. There is one floating node in each circuit and the goal is to find the best location for it. We have implemented two algorithms: The mass center approach proposed in [PB91] and the MFSPT based algorithm proposed in this paper. The cost of the final set of Steiner trees is approximated by the cost of the corresponding minimum-spanning trees. We have found that the MFSPT based approach produces better results as compared with the the mass center approach in every case. The average improvement is 1.7%.

In the second class of experiments, two layout-driven logic synthesis algorithms are compared. The first one is base on the MFSPT (as described in the paper) and the second one is based on the mass center trees (as described in [PB91]). For fair comparison, we have used the same logic factorization algorithm (SIS—a CAD tool from UC, Berkley) in both algorithms. Input and output pads are fixed on the boundary, while the gates can be placed anywhere to reduce the total length. The results of MCNC benchmarks are summarized in Table 1. The third column shows the results without length minimization. The fourth column shows the results of mass center approach, and the last column demonstrates the results of the MFSPT algorithm.

Two examples are shown in Figures 2, 3. In Figure 2, the circuit is a small one generated by us (it is actually a subcircuit of benchmark misex1). Figure 3 shows the results of circuit z4ml. Each vertex shown is either an AND or an OR gate.

## 5 Conclusion

We formulated a new placement problem that find applications, for example, in layout-driven logic synthesis. As a basis for solving the problem we studied the minimum floating Steiner tree (MFST) problem and proposed two approximation algorithms with provable bounds. An extension of the MFST problem, called the minimum floating tree problem, has also been studied. By decomposing trees to non-crossing segments, we generalized the idea of Dirichlet partition from nodes to line segments.

Experimental results show that the new formulation produces results better than previously proposed technique (of [PB91]).

Currently we are studying the possibility of incorporating other objective functions in the above framework, in particular, timing and low-power issues. To address timing-driven designs we are considering to incorporate (a generalization of) the algorithm proposed in [CKR+92]. To address power consumption, in the above formulation, active nets (i.e, nets with high transition density) are assigned large weights. Then, our goal is to minimize a weighted length of the nets. Thus, to address this problem we need to understand weighted floating Steiner trees (by generalizing the concept of weighted Steiner trees introduced in [CWS92]).

## References


[CCMS93] D. I. Cheng, S. Chang, and M. Marek-Sadowska. "Partitioning Combinational Circuits in Graph and Logic Domains". In *Proceedings of Synthesis And SImulation Meeting and International Interchange (SASIMI-93)*, 1993.

[CKR+92] J. Cong, A. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. "Provably Good Performance-Driven Global Routing". *IEEE Transactions on Computer Aided Design*, 11(6):739–752, June 1992.

[CWS92] C. Chiang, C. K. Wong, and M. Sarrafzadeh. "A Globar Router Based on Weighted Steiner Trees". Technical report, Northwestern University, 1992. to appear in IEEE Transactions on Computer -Aided Design, 1994.

[GP87] G. Georgakopoulos and C. H. Papadimitriou. "The 1-Steiner Tree Problem". *Journal of Algorithms*, 8:122–130, 1987.

[KR92] A. Kahng and G. Robins. "A New Class of Iterative Steiner Tree Heuristics with Good Performance". *IEEE Transactions on Computer Aided Design*, 11(7):893–902, 1992.

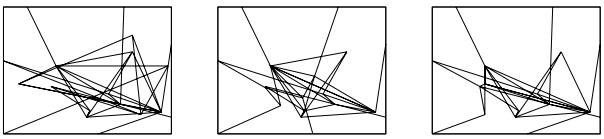
[LPP93] S. Liu, K. Pan, and M. Pedram. "Alleviating Routing Congestion by Combining Logic Resynthesis and Linear Placement". In *European Design Automation Conference*, pages 578–582, 1993.

[PB91] M. Pedram and N. Bhat. "Layout Driven Technology Mapping". In *Design Automation Conference*, pages 99–105, 1991.

[SM91] K. Shahookar and P. Mazumder. "VLSI Cell Placement Techniques". *ACM Computing Surveys*, 23(2):143–220, June 1991.

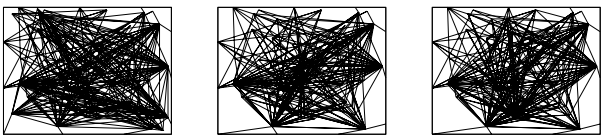

| Circuit | Gates | Original length | MC length | Floating length |
|---|---|---|---|---|
| misex1 | 51 | 620 | 483 | 465 |
| z4ml | 87 | 950 | 855 | 792 |
| misex2 | 100 | 1255 | 995 | 946 |
| count | 146 | 1773 | 1552 | 1502 |
| 5xp1 | 159 | 1917 | 1611 | 1574 |
| b9 | 159 | 1924 | 1582 | 1555 |
| 9symml | 165 | 1977 | 1682 | 1659 |
| apex7 | 228 | 2713 | 2474 | 2336 |
| 9sym | 251 | 3096 | 2660 | 2627 |
| rd73 | 277 | 3384 | 2896 | 2825 |
| alu2 | 282 | 3466 | 2973 | 2911 |
| C880 | 362 | 4453 | 3995 | 3868 |
| C499 | 411 | 5079 | 4678 | 4543 |
| Improvement | | | 14.1% | 16.9% |

Table 1: Results of the benchmark circuits.



(a) (b) (c)

Figure 2: (a) An initial layout of the small circuit. (b) The layout after we apply the mass center approach. (c) The layout after we apply the MFSPT based algorithm.



(a) (b) (c)

Figure 3: (a) An initial layout of circuit z4ml. (b) The layout after we apply the mass center approach. (c) The layout after we apply the MFSPT based algorithm.