SafeChoice: A Novel Clustering Algorithm for Wirelength-Driven Placement*

Jackey Z. Yan
Department of ECE
Iowa State University
Ames, IA 50010 USA
zijunyan@iastate.edu

Chris Chu
Department of ECE
Iowa State University
Ames, IA 50010 USA
cnchu@iastate.edu

Wai-Kei Mak
Department of CS
National Tsing Hua University
Hsingchu 300, Taiwan
wkmak@cs.nthu.edu.tw

ABSTRACT

This paper presents SafeChoice (SC), a novel clustering algorithm for wirelength-driven placement. Unlike all previous approaches, SC is proposed based on a fundamental theorem, *safe condition* which guarantees that clustering would not degrade the placement wirelength. To derive such a theorem, we first introduce the concept of *safe clustering*, i.e., do clustering without degrading the placement quality. To check the safe condition for pair-wise clustering, we propose selective enumeration technique. SC maintains a global priority queue (PQ) based on the safeness and area of potential clusters. Iteratively the cluster at the top of the PQ is formed. SC automatically stops clustering when generating more clusters would degrade the placement wirelength. To achieve other clustering objectives, e.g., any target clustering ratio, SC is able to perform under three different modes. Comprehensive experimental results show that the clusters produced by SC consistently help the placer to achieve the best wirelength among all other clustering algorithms.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits, Design Aids**]: Placement and routing

General Terms

Algorithms, Design, Performance

Keywords

Hypergraph Clustering, VLSI Placement, Physical Design

1. INTRODUCTION

For modern VLSI designs, placement is the most critical stage in the physical synthesis flow. It has significant impacts on timing, routing and even manufacturing. In the nanometer scale era, a circuit typically contains millions of objects. It is extremely challenging for a modern placer to be reasonably fast, yet still be able to produce good solutions. Clustering cuts down the problem size via combining highly connected objects, so that the placers can perform more efficiently and effectively on a smaller problem. It is an attractive solution to cope with the ever-increasing design complexity. Therefore, as an essential approach to improve both the runtime and quality of result, various clustering algorithms have been adopted in the state-of-the-art placement algorithms [1–7].

*This work was partially supported by IBM Faculty Award, NSF under grant CCF-0540998 and NSC under grant NSC 98-2220-E-007-031.

ISPD'10, March 14–17, 2010, San Francisco, California, USA.

1.1 Previous Work

Clustering is a traditional problem in VLSI CAD area. The clustering algorithms proposed long time ago were described in [8]. In the last several years, various new algorithms were proposed to continue improving the clustering quality. In [9] Karypis et al. proposed edge coarsening (EC) clustering. In EC objects are randomly visited. Each object is clustered with the most highly-connected unvisited neighbor object. The connectivity between two objects is computed as the total weight of all edges connecting them with hyperedges represented by a clique model. FirstChoice (FC) clustering was developed in [10] and is very similar to EC. The only difference between them is that for each object in FC, all of its neighbor objects are considered for clustering. FC has been used in placers NTUplace3 [1] and Capo [2]. However, neither EC nor FC considers the impact of cluster size on the clustering quality. Alpert et al. [11] and Chan et al. [12] improved EC and FC respectively, by considering the area of clusters, i.e., clusters with smaller area are preferred to be generated. Cong et al. [13] proposed an edge separability-based clustering (ESC). Unlike previous methods, ESC uses edge separability to guide the clustering process. To explore global connectivity information, all edges are ranked via a priority queue (PQ) based on the edge separability. Without violating the cluster size limit, the two objects in the highest ranking edge are clustered. Hu et al. [14] developed fine granularity (FG) clustering. The difference between FG and ESC is that for FG the order in the PQ is based on edge contraction measured by a mutual contraction metric. FG has been used in placer mFAR [3]. Nam et al. [15] proposed BestChoice (BC) clustering which has been widely used in the top-of-the-line placers APlace [4], mPL6 [5], FastPlace3 [6], and RQL [7]. Instead of ranking the edges, BC maintains a PQ based on a pair of objects, i.e., each object and its best neighbor object. A score function considering both hyperedge weight and object area is derived to calculate the score between two objects. For each object, the best neighbor object is the neighbor object with the highest score. The two objects at the top of the PQ are clustered iteratively. But updating such a PQ is quite time-consuming. Hence, the authors proposed a lazy-update technique to make a trade-off between the clustering runtime and quality.

All of the above clustering algorithms either explicitly or implicitly transform a hyperedge into a clique model, so that they can handle pair-wise clustering, i.e., cluster two objects at each time. Recently, Li et al. [16] presented NetCluster (NC) that can handle hyperedges directly and cluster more than two objects at one time. In NC, initial clusters are first generated by FM algorithm. Then a score is assigned to each net. The objects in the net with the highest score are clustered.

For all previous clustering algorithms, none of them specifically aims at improving the placement quality. They proposed a variety of heuristics, e.g., different score functions, to measure the connectivity among the objects, so that the most highly-connected objects are clustered. Of course, the benefit is a reduction of problem size. But, clustering such objects may not help the placer to produce better solution. This is because clustering forces some objects to stay together during placement, which constrains the solution space exploration of the placer. If such constraint is enforced improperly, i.e., cluster objects that should not be clustered, the placement solution would be jeopardized. It has not been proved that highly-connected objects should be clustered. So we believe the fundamental problem of clustering for placement is: *How to do clustering, so that it can be guaranteed that clustering would not degrade the placement quality?*

1.2 Our Contributions

In this work, we present a novel clustering algorithm called SafeChoice (SC) for wirelength-driven placement. SC handles hyperedges directly. Different from all previous clustering algorithms, SC is proposed based on a fundamental theorem, which guarantees that clustering would not degrade the placement quality. None of previous techniques has such guarantee. Additionally, three operation modes of SC are presented to achieve various clustering objectives. Essentially, we have five main contributions:

- **Concept of Safe Clustering**: We introduce the concept of *safe clustering*. If clustering some objects would not degrade the wirelength in an optimal placement, it is safe to cluster such objects.
- **Safe Condition**: Based on the concept of safe clustering, we derive the fundamental theorem — *safe condition* for pair-wise clustering. We prove that if any two objects satisfy the safe condition, clustering them would not degrade the wirelength.
- **Selective Enumeration**: To check the safe condition for pair-wise clustering, we propose selective enumeration. With this method, we can efficiently find out the safe clusters in a circuit.
- **SafeChoice**: We present SafeChoice algorithm that globally ranks potential clusters via a PQ based on their safeness and area. Iteratively the cluster at the top of the PQ will be formed.
- **Smart Stopping Criterion**: A smart stopping criterion is proposed based on a simple heuristic. So it can automatically stop clustering once generating more clusters would start to degrade the placement wirelength. As far as we know, none of previous algorithms has such feature.

We compare SC with three state-of-the-art clustering algorithms FC, BC and NC. The results show that the clusters produced by SC consistently helps the placer to generate the best wirelength.

The rest of this paper is organized as follows. Section 2 describes the safe clustering. Section 3 introduces the algorithm of SafeChoice. Experimental results are presented in Section 4. Finally, this paper ends with a conclusion and the direction of future work.

2. SAFE CLUSTERING

In this section, we first introduce the concept of safe clustering. Then based on this concept we derive the safe condition for pair-wise clustering. Finally we propose selective enumeration to practically check the safe condition for any two objects in the circuit.

First of all, we introduce some notations used in the discussion. The original netlist is modeled by a hypergraph $G(V, E)$, where V is the set of vertices and E is the set of hyperedges. Given $v \in V$, E_v is the set of hyperedges incident to v, and $\overline{E}_v = E - E_v$. Let P be the set of all possible placements of the vertices in V. The wirelength is measured by weighted Half-Perimeter Wirelength (HPWL).

2.1 Concept of Safe Clustering

The concept of safe clustering is defined as follows.

DEFINITION 1. **Safe Clustering:** *For a set of vertices $V_c \subseteq V$ ($|V_c| \geq 2$), if the optimal wirelength of the netlist generated by clustering V_c is the same as the optimal wirelength of the original netlist, then it is safe to cluster the vertices in V_c.*

The placement problem is NP-hard. In practice we cannot find the optimal wirelength for a real circuit. So we present a more practical definition below.

DEFINITION 2. **Safe Clustering*:** *$\forall p \in P$, if a set of vertices $V_c \subseteq V$ ($|V_c| \geq 2$) can be moved to the same location without increasing the wirelength, then it is safe to cluster the vertices in V_c. (Assume the total area of vertices in V_c is very small, so that we can ignore the overlap issue.)*

Definition 2 is stronger than Definition 1. If V_c is safe for clustering based on Definition 2, it is also safe under Definition 1. In the rest of this paper, we employ Definition 2 for discussion.

Based on Definition 2, we derive the definitions for horizontally and vertically safe clustering as follows.

DEFINITION 3. **Horizontally/Vertically Safe Clustering:** *$\forall p \in P$, if a set of vertices $V_c \subseteq V$ ($|V_c| \geq 2$) can be horizontally/vertically moved to the same x/y coordinate without increasing the wirelength in x/y direction, then it is horizontally/vertically safe to cluster the vertices in V_c. (Assume the total area of vertices in V_c is very small, so that we can ignore the overlap issue.)*

Now we show that if vertices in V_c are both horizontally and vertically safe for clustering, then it is safe to cluster them under Definition 2. Given any initial placement $p \in P$, firstly we move those vertices horizontally to the same x coordinate. Secondly, we move them vertically to the same y coordinate. Consequently, the vertices in V_c are moved to the same location. Based on Definition 3 the wirelength would not increase during the movements. So it is safe to cluster the vertices in V_c by Definition 2.

Without considering fixed vertices, e.g., I/O objects, if vertices in V_c are horizontally safe for clustering, then they are always vertically safe for clustering as well. This is because a vertical movement in a placement p is the same as a horizontal movement in another placement obtained by rotating p by 90°. If a set of vertices is horizontally safe for clustering, it is also vertically safe for clustering. So for the following discussion it is sufficient to consider only x direction.

2.2 Safe Condition for Pair-Wise Clustering

From Definition 2 we derive a condition to mathematically determine whether it is safe to cluster the vertices in V_c. Firstly, we define two key functions for the derivation. For the sake of simplicity, we always assume V_c contains only two vertices a and b, i.e., $V_c = \{a, b\}$.

DEFINITION 4. **Wirelength Gradient Function:** *Given a placement $p \in P$ and a hyperedge $e \in E$, assuming a is on the left of b, we define*

$\Delta_a(p, e)$: *Gradient function of wirelength of e if a is moving towards b.*
$\Delta_b(p, e)$: *Gradient function of wirelength of e if b is moving towards a.*

Let $w_e (w_e \geq 0)$ be the weight of e. From Definition 4 we have

$$\Delta_a(p,e) = \begin{cases} w_e & \text{if } a \text{ is the rightmost vertex of } e \\ -w_e & \text{if } a \text{ is the } \textit{only} \text{ leftmost vertex of } e \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta_b(p,e) = \begin{cases} w_e & \text{if } b \text{ is the leftmost vertex of } e \\ -w_e & \text{if } b \text{ is the } \textit{only} \text{ rightmost vertex of } e \\ 0 & \text{otherwise} \end{cases}$$

Considering a is moving towards b in p, if $\Delta_a(p, e) > 0$, it means the wirelength of e will increase; if $\Delta_a(p, e) < 0$, then the wirelength of e will decrease; otherwise the wirelength of e will not change.

DEFINITION 5. **Total wirelength Gradient Function:** *Given a placement $p \in P$ and $V_c = \{a, b\}$, we define*

$$\mathcal{F}_{ab}(p) = min(\sum_{e \in E_a} \Delta_a(p, e), \sum_{e \in E_b} \Delta_b(p, e))$$

In p if both a and b move towards each other, $\mathcal{F}_{ab}(p)$ first calculates the total wirelength change of all hyperedges for moving a and b, respectively. Then it returns the one with smaller change. For example, if $\mathcal{F}_{ab}(p) = \sum_{e \in E_a} \Delta_a(p, e) \leq 0$, it means moving a towards b would not increase the total wirelength; if $\mathcal{F}_{ab}(p) > 0$, then moving either a or b towards each other would increase the total wirelength. Next, we use this function to derive the safe condition for a and b.

THEOREM 1. **Safe Condition** *for $V_c = \{a, b\}$*

It is safe to cluster a and b if $\forall p \in P, \mathcal{F}_{ab}(p) \leq 0$

PROOF. Given an initial placement $p^0 \in P$ with total wirelength l^0. Because $\forall p \in P, \mathcal{F}_{ab}(p) \leq 0$, we have $\mathcal{F}_{ab}(p^0) \leq 0$. Suppose $\mathcal{F}_{ab}(p^0) = \sum_{e \in E_a} \Delta_a(p^0, e) \leq 0$. This means by moving a a small distance towards b, the total wirelength of all hyperedges would not increase. After such movement, we get another placement p^1 with total wirelength l^1, where $l^0 \geq l^1$. For p^1 we still have $\mathcal{F}_{ab}(p^1) \leq 0$. Suppose this time $\mathcal{F}_{ab}(p^1) = \sum_{e \in E_b} \Delta_b(p^1, e) \leq 0$. This means moving b a small distance towards a would not increase the total wirelength. Again, after such movement, we get another placement p^2 with total wirelength l^2, where $l^1 \geq l^2$. We keep moving either a or b towards each other until they reach the same location. Suppose the final total wirelength is l^n. Because after each movement we always have $\mathcal{F}_{ab}(p) \leq 0$, which means the total wirelength would not increase, eventually we have $l^0 \geq l^n$.

As a result, given any initial placement p^0 we can gradually move a and b to the same location without increasing the wirelength. So based on Definition 2, it is safe to cluster vertices a and b. □

2.3 Selective Enumeration

To check whether it is safe to cluster a and b, Theorem 1 shows that we need to generate all placements in P. To do so, we have to enumerate all possible positions for all vertices in V. Apparently this is not a practical approach. In this section, we show that in order to check Theorem 1, it is sufficient to consider only a small subset of placements. Selective enumeration technique is proposed to enumerate such necessary placements.

Selective enumeration is motivated by the following principle: *Given two placements $p_1, p_2 \in P$, if we know $\mathcal{F}_{ab}(p_1) \leq \mathcal{F}_{ab}(p_2)$, then p_1 can be ignored in the enumeration*. This is because Theorem 1 shows that the safe condition is only determined by the placement with the maximum $\mathcal{F}_{ab}(p)$ value. So the basic idea of selective enumeration is to find out the relationship of $\mathcal{F}_{ab}(p)$ values among different placements, so that in the enumeration process we can ignore the placements with smaller or equal $\mathcal{F}_{ab}(p)$ values. Placements in P are generated by different positions of different vertices. Our goal is to identify some vertices in V, such that some or even all of their possible positions can be ignored.

We first classify the vertices in V into two categories $V_{\bar{a}\bar{b}}$ and V_{ab} ($V_{\bar{a}\bar{b}} \cup V_{ab} \cup \{a, b\} = V$). Then we discuss the enumeration of their positions separately. $\forall v \in V$, x_v denotes the x coordinate of v.

1. $V_{\bar{a}\bar{b}}$: vertices connecting with neither a nor b.
2. V_{ab}: vertices connecting with at least one of a and b.

LEMMA 1. *Given a placement $p \in P$, by moving vertex $v \in V_{\bar{a}\bar{b}}$ to any other position, another placement $p' \in P$ is generated. We have $\mathcal{F}_{ab}(p) = \mathcal{F}_{ab}(p')$.*

PROOF. Since $\forall v \in V_{\bar{a}\bar{b}}$, v connects with neither a nor b, changing the position of v would not change the leftmost or rightmost vertex of any hyperedge connecting with a or b. Therefore,

$$\forall e \in E_a,\ \Delta_a(p, e) = \Delta_a(p', e)$$
$$\forall e \in E_b,\ \Delta_b(p, e) = \Delta_b(p', e)$$

Thus, $\mathcal{F}_{ab}(p) = \mathcal{F}_{ab}(p')$. □

Based on Lemma 1, in the enumeration we can simply ignore all vertices in $V_{\bar{a}\bar{b}}$.

LEMMA 2. *Given a placement $p \in P$, vertex $v \in V_{ab}$ and $x_v = k_1$. After moving v to $x_v = k_2$, another placement $p' \in P$ is generated. We have $\mathcal{F}_{ab}(p) = \mathcal{F}_{ab}(p')$ if any one of the following conditions is satisfied: (1) $k_1 \leq x_a$ and $k_2 \leq x_a$; (2) $k_1 \geq x_b$ and $k_2 \geq x_b$; (3) $x_a < k_1 < x_b$ and $x_a < k_2 < x_b$.*

PROOF. Suppose condition (1) holds, i.e., v is on the left of a in both p and p'. $\forall e \in E_v$, we consider two[1] possible values of $\Delta_a(p, e)$:

- $\Delta_a(p, e) = w_e$
  This means a is the rightmost vertex of e in p. After moving v to k_2, because $k_2 \leq x_a$, a is still the rightmost vertex of e in p'. Thus, $\Delta_a(p', e) = w_e = \Delta_a(p, e)$.
- $\Delta_a(p, e) = 0$
  This means a is neither the only leftmost nor the rightmost vertex of e in p. After moving v to k_2, because $k_2 \leq x_a$, v is still on the left of a in p'. Thus, $\Delta_a(p', e) = 0 = \Delta_a(p, e)$.

So $\forall e \in E_v$, $\Delta_a(p, e) = \Delta_a(p', e)$. Similarly we have $\forall e \in E_v$, $\Delta_b(p, e) = \Delta_b(p', e)$. Therefore,

$$\forall e \in E_a, \Delta_a(p, e) = \Delta_a(p', e)$$
$$\forall e \in E_b, \Delta_b(p, e) = \Delta_b(p', e)$$

Thus, $\mathcal{F}_{ab}(p) = \mathcal{F}_{ab}(p')$. Analogically, the cases for conditions (2) and (3) can be proved as well. □

Lemma 2 shows that $\forall v \in V_{ab}$, instead of enumerating all possible positions, we only need to consider three possibilities: (1) v is on the left of a ($x_v \leq x_a$); (2) v is on the right of b ($x_v \geq x_b$); (3) v is between a and b ($x_a < x_v < x_b$).

Based on Lemma 1 and 2, we need to enumerate $3^{|V_{ab}|}$ different placements rather than all placements in P. Next, we will further cut down this number from $3^{|V_{ab}|}$ to $2^{|V_{ab}|}$, by ignoring all positions between a and b.

LEMMA 3. *Given a placement $p \in P$, such that vertex $v \in V_{ab}$ is between a and b ($x_a < x_v < x_b$). After moving v either to the left of a or to the right of b, another placement $p' \in P$ is generated. We have $\mathcal{F}_{ab}(p) \leq \mathcal{F}_{ab}(p')$.*

PROOF. Suppose v is moved to the left of a.
For a, after the movement, a might become the rightmost vertex of some hyperedge. So we have

$$\forall e \in E_v, \Delta_a(p, e) \leq \Delta_a(p', e) \quad (1)$$

For b, after the movement, v is still on the left of b. So we have

$$\forall e \in E_v, \Delta_b(p, e) = \Delta_b(p', e) \quad (2)$$

Based on Equations 1–2, we have

$$\forall e \in E_a, \Delta_a(p, e) \leq \Delta_a(p', e)$$
$$\forall e \in E_b, \Delta_b(p, e) = \Delta_b(p', e)$$

Thus, $\mathcal{F}_{ab}(p) \leq \mathcal{F}_{ab}(p')$. Similarly, we can prove the case for v is moved to the right of b. □

[1] Because v is on the left of a, a would not become the only leftmost vertex of e. Thus, $\Delta_a(p, e) \neq -w_e$.

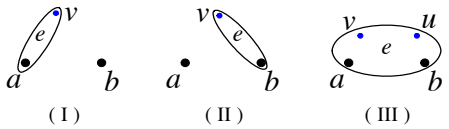


Figure 1: Simple examples of vertices that can be fixed.

So far, we have proved that we only need to consider two possible positions (on the left of a and on the right of b) for each vertex in V_{ab}, i.e., totally $2^{|V_{ab}|}$ different placements. In a modern circuit, $|V_{ab}|$ may become more than 1000. So practically $2^{|V_{ab}|}$ is still too big to enumerate. Therefore, we intend to further cut down this number.

We notice that for some vertices in V_{ab}, it is not always necessary to consider both of the two possible positions. For example in Fig. 1-(I), v is only connected with a via e. If v is on the left of a in placement p_l, then $\mathcal{F}_{ab}(p_l) = min(w_e, 0) = 0$; if v is on the right of b in placement p_r, then $\mathcal{F}_{ab}(p_r) = min(-w_e, 0) = -w_e$. We have $\mathcal{F}_{ab}(p_l) > \mathcal{F}_{ab}(p_r)$. So, we can ignore p_r where v is on the right of b. To make use of such property and further reduce the enumeration size, in the following part we identify three subsets of vertices in V_{ab} (V^I, V^{II} and V^{III}), and prove that under certain condition the positions of those vertices can be fixed in the enumeration.

I. $V^I = \{v | v \in V_{ab}$ **and** $\exists e \in (E_a \cap \overline{E}_b)$ **s.t.** $v \in e$ **and** $\nexists e \in (E_a \cap E_b)$ **s.t.** $v \in e\}$
(e.g., in Fig. 1-(I) vertex $v \in V^I$)

II. $V^{II} = \{v | v \in V_{ab}$ **and** $\exists e \in (\overline{E}_a \cap E_b)$ **s.t.** $v \in e$ **and** $\nexists e \in (E_a \cap E_b)$ **s.t.** $v \in e\}$
(e.g., in Fig. 1-(II) vertex $v \in V^{II}$)

III. $V^{III} = \{v | v \in V_{ab}$ **and** $\nexists e \in (\overline{E}_a \cap E_b)$ **s.t.** $v \in e$ **and** $\nexists e \in (E_a \cap \overline{E}_b)$ **s.t.** $v \in e$ **and** $\exists e \in (E_a \cap E_b)$ **s.t.** $v \in e\}$
(e.g., in Fig. 1-(III) vertices $v, u \in V^{III}$)

LEMMA 4. *Given a placement $p \in P$, such that vertex $v \in V^I$ is on the left of a. After moving v to the right of b, another placement $p' \in P$ is generated. We have $\mathcal{F}_{ab}(p) \geq \mathcal{F}_{ab}(p')$.*

PROOF. Let $E_{v \cap ab} = E_v \cap (E_a \cup E_b)$.

- In placement $p, \forall e \in E_{v \cap ab}$, we consider two cases:
  - $\exists$ vertex $c \in e (c \neq a, c \neq v)$, s.t. $x_c \geq x_b$
    Because $x_v \leq x_a$ and $x_c \geq x_b, x_v \leq x_a \leq x_c$. a is neither the only leftmost nor the rightmost vertex of e. So $\Delta_a(p, e) = 0$.
  - $\nexists$ vertex $c \in e (c \neq a, c \neq v)$, s.t. $x_c \geq x_b$
    Because $x_v \leq x_a$ and no other vertices in e are on the right of b, a is the rightmost vertex of e. So $\Delta_a(p, e) = w_e$.

  Thus, $\forall e \in E_{v \cap ab}, \Delta_a(p, e) \geq 0$.
- In placement $p', \forall e \in E_{v \cap ab}$, we consider two cases:
  - $\exists$ vertex $c \in e (c \neq a, r \neq v)$, s.t. $x_c \leq x_a$
    Because $x'_v \geq x_b$ and $x_c \leq x_a, x_c \leq x_a \leq x'_v$. a is neither the only leftmost nor the rightmost vertex of e. So $\Delta_a(p', e) = 0$.
  - $\nexists$ vertex $c \in e (c \neq a, c \neq v)$, s.t. $x_c \leq x_a$
    Because $x'_v \geq x_b$ and no other vertices in e are on the left of a, a is the only leftmost vertex of e. So $\Delta_a(p, e) = -w_e$.

  Thus, $\forall e \in E_{v \cap ab}, \Delta_a(p', e) \leq 0$.

So $\forall e \in E_{v \cap ab}, \Delta_a(p, e) \geq \Delta_a(p', e)$. Also $\forall v \in V^I$, v does not connect with b, so $\forall e \in E_{v \cap ab}, \Delta_b(p, e) = \Delta_b(p', e)$. Therefore,

$$\forall e \in E_a, \Delta_a(p, e) \geq \Delta_a(p', e)$$
$$\forall e \in E_b, \Delta_b(p, e) = \Delta_b(p', e)$$

Thus, $\mathcal{F}_{ab}(p) \geq \mathcal{F}_{ab}(p')$. □

From Lemma 4, $\forall v \in V^I$ we can fix v on the left of a. As V^{II} is symmetrical with V^I, similarly we can prove that $\forall v \in V^{II}$ we can fix v on the right of b.

LEMMA 5. *Given a placement $p \in P$, such that vertex $v \in V^{III}$ is on the left of a, vertex $u \in V^{III}$ is on the right of b, and $E_v \cap (E_a \cup E_b) = E_u \cap (E_a \cup E_b)$. After moving either one or both of them to another position, i.e., moving v to the right of b and u to the left of a, another placement p' is generated. We have $\mathcal{F}_{ab}(p) \geq \mathcal{F}_{ab}(p')$.*

PROOF. Let $E_{v-u} = E_v \cap (E_a \cup E_b) = E_u \cap (E_a \cup E_b)$. We consider all three possible movements of v and u.

- v **moved to the right of** b**,** u **did not move**
  In placement $p', \forall e \in E_{v-u}$ we consider two cases:
  - $\exists$ vertex $c \in e (c \neq a)$, s.t. $x_c \leq x_a$
    In this case, a is neither the only leftmost nor the rightmost vertex in e, and b is neither the leftmost nor the only rightmost vertex in e. So $\Delta_a(p', e) = 0, \Delta_b(p', e) = 0$.
  - $\nexists$ vertex $c \in e (c \neq a)$, s.t. $x_c \leq x_a$
    In this case, a is the only leftmost vertex in e, and b is neither the leftmost nor the only rightmost vertex in e. So $\Delta_a(p', e) = -w_e, \Delta_b(p', e) = 0$.
- u **moved to the left of** a**,** v **did not move**
  In placement $p', \forall e \in E_{v-u}$ we consider two cases:
  - $\exists$ vertex $c \in e (c \neq b)$, s.t. $x_c \geq x_b$
    In this case, a is neither the only leftmost nor the rightmost vertex in e, and b is neither the leftmost nor the only rightmost vertex in e. So $\Delta_a(p', e) = 0, \Delta_b(p', e) = 0$.
  - $\nexists$ vertex $c \in e (c \neq b)$, s.t. $x_c \geq x_b$
    In this case, b is the only rightmost vertex in e, and a is neither the only leftmost nor the rightmost vertex in e. So $\Delta_a(p', e) = 0, \Delta_b(p', e) = -w_e$.
- v **moved to the right of** b**,** u **moved to the left of** a
  In this case, a is neither the only leftmost nor the rightmost vertex in e, and b is neither the leftmost nor the only rightmost vertex in e. So $\forall e \in E_{v-u}, \Delta_a(p', e) = 0, \Delta_b(p', e) = 0$.

For all of the above cases, $\Delta_a(p', e) \leq 0$ and $\Delta_b(p', e) \leq 0$. In placement $p, \forall e \in E_{v-u}$ because a is neither the only leftmost nor the rightmost vertex in e, and b is neither the leftmost nor the only rightmost vertex in e, we have $\Delta_a(p, e) = \Delta_b(p, e) = 0$. As a result, we have $\forall e \in E_{v-u}, \Delta_a(p, e) \geq \Delta_a(p', e), \Delta_b(p, e) \geq \Delta_b(p', e)$, Therefore,

$$\forall e \in E_a, \Delta_a(p, e) \geq \Delta_a(p', e)$$
$$\forall e \in E_b, \Delta_b(p, e) \geq \Delta_b(p', e)$$

Thus, $\mathcal{F}_{ab}(p) \geq \mathcal{F}_{ab}(p')$. □

Lemma 5 shows that if $\exists v, u \in V^{III}$ and $E_v \cap (E_a \cup E_b) = E_u \cap (E_a \cup E_b)$, then we can fix v to the left of a and u to the right of b.

In all, we have identified three subsets of vertices in V_{ab}. If certain condition is satisfied, those vertices can be fixed in the enumeration. Note that those three subsets may not include all vertices that can be fixed in V_{ab}. We believe more complicated subsets and conditions can be derived. But for the sake of simplicity, SafeChoice considers only the above three subsets.

Let the total number of vertices in V^I, V^{II} and V^{III} be α. As a result, given two objects a and b, we only need to enumerate $L = 2^{|V_{ab}| - \alpha}$ different placements. To limit runtime, at most 2^{10} placements are enumerated by default. If $|V_{ab}| - \alpha > 10$, we simply would not consider clustering a and b [2]. For each of those enumerated placement p_i ($1 \leq i \leq L$), we calculate a score $s_i = \mathcal{F}_{ab}(p_i)$. We define $s_{max} = max(s_1, s_2, ..., s_L)$. Based on Theorem 1, if $s_{max} \leq 0$, then it is safe to cluster a and b.

[2] By considering the three subsets, we have $|V_{ab}| - \alpha \leq 10$ for most pairs in practice.

Table 1: Differences among three modes (SC is the default mode).

Mode	Clustering Objective	S^*	Stopping Criterion
SC-G	safe clusters guarantee	s_{max}	no more safe clusters is in PQ
SC-R	target clustering ratio	$\overline{s}$	target clustering ratio is reached
SC	best placement wirelength	$\overline{s}$	threshold cost C_t is reached

3. ALGORITHM OF SAFECHOICE

In the previous section, we have described a practical method of checking the safe condition for pair-wise clustering. Here, we apply this method in a PQ-based algorithm flow and propose SafeChoice algorithm. To satisfy various clustering objectives, we present three operation modes for SafeChoice.

3.1 Priority-Queue Based Framework

Previous work [11, 12] show that the cluster size has significant impacts on the clustering quality. If two potential clusters have the same connectivity information, the one with the smaller area is preferred to be formed first. So in SafeChoice to balance the safeness and area, we use the following cost function to calculate the cost C for clustering two objects a and b.

$$C(a,b) = S^* + \theta \times \frac{A_a + A_b}{\overline{A}_s} \tag{3}$$

where $\theta = 4$ by default, A_a and A_b denote the area of a and b respectively, $\overline{A}_s$ is the average standard cell area in a circuit, and S^* is a term describing the safeness of clustering a and b. S^* is calculated based on different modes of SafeChoice (see Section 3.2).

In SafeChoice we maintain a global PQ similar to that in [15]. But we rank each pair of objects based on the cost obtained by Equation 3. For SafeChoice, it is time-consuming to consider all possible pairs in V. So for each object, we only consider its neighbor objects connected by the nets containing at most β objects ($\beta = 7$ by default). Iteratively, SafeChoice clusters the pair of objects at the top of the PQ, and then update the PQ using lazy-update. For different operation modes, SafeChoice stops clustering based on different stopping criteria, which will be addressed in Section 3.2.

3.2 Operation Modes of SafeChoice

Given a circuit, some algorithms (e.g., FC and BC) can reach any clustering ratio γ [3], while others (e.g., FG and NC) can only reach a certain γ. None of previous work is able to automatically stop clustering when the γ for the best placement wirelength is reached. By default SafeChoice automatically stops clustering when generating more clusters would degrade the placement wirelength. Additionally, to achieve other clustering objectives, e.g., any target γ, SafeChoice is capable of performing under various modes (see Table 1):

- **Safety Guarantee Mode [SC-G]**
  SC-G aims at producing the completely safe clusters. Under this mode, $S^* = s_{max}$ in Equation 3. In each iteration, we cluster the pair of objects at the top of the PQ only if its $S^* \leq 0$. Based on Theorem 1, we guarantee that the formed clusters are safe. SC-G terminates when there is no such safe clusters in the PQ.
- **Clustering Ratio Mode [SC-R]**
  The SC-G mode may not achieve low clustering ratio in practice, because the number of safe clusters in a circuit is usually limited. Sometimes if clustering cannot significantly reduce the circuit size, even though all clusters are safe, the placer may not perform efficiently and produce better result. So to make a trade-off between safeness and circuit size reduction, SC-R produces some unsafe clusters, besides the safe ones. We derive the following function to evaluate the safeness of each cluster:

$$\overline{s} = \frac{\sum_{i=1}^{L} s_i}{L} \tag{4}$$

[3]The *clustering ratio* γ is defined as the ratio of the number of objects in the clustered circuit to the number of objects in the original circuit.

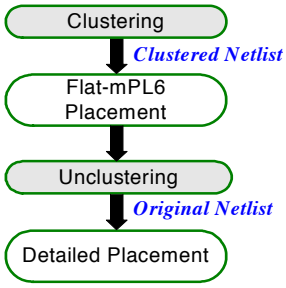


Figure 2: Experimental flow for clustering algorithm.

Basically, for a pair of objects a and b Equation 4 calculates the average score $\overline{s}$ over the L enumerated placements. Under SC-R mode, $S^* = \overline{s}$ in Equation 3. Iteratively, SC-R clusters the pair of objects at the top of the PQ until the target γ is reached.

- **Smart Mode [SC] (default mode)**
  The smart mode heuristically stops the clustering process when a typical placer achieves the best placement wirelength. None of previous clustering algorithms is able to do this. For different circuits, the γ for the best placement wirelength may be different. In SC, we set a threshold cost C_t ($C_t = 21$ by default), and use the same cost function as in SC-R. During the clustering process, SC would not terminate until the cost reaches C_t. With this simple heuristic, SC is able to automatically stop when generating more clusters starts to degrade the placement wirelength.

4. EXPERIMENTAL RESULTS

Experiments are run on a Linux server with Intel Xeon 2.83 GHz CPU and 32 GB memory. We compare SC with three clustering algorithms FC [12], BC [15] and NC [16]. We implemented FC and BC by ourselves and obtained the binary of NC from the authors in [16]. For BC the lazy-update [15] is used to speed up its runtime. ISPD 05/06 placement benchmarks [17,18] are used as the test circuits.

In the experiments, the clustering algorithm is applied as a pre-processing step before placement (see Fig. 2). We adopt mPL6 [5] as the placer. mPL6 is based on a multilevel framework, and uses BC as its internal clustering algorithm. Without turning off BC inside mPL6, we cannot fairly compare the effectiveness of various clustering algorithms, because the internal clustering process will produce some noise to the results. So we add "-cluster_ratio 1" to the command line, such that mPL6 performs only one-level placement without any clustering inside, i.e., flat-mPL6. *As far as we know, mPL6 is the only placer that can turn off the internal clustering without modifying the source code*. In Fig. 2 after unclustering, we arrange the objects inside each cluster in one row. The order among those objects are random. Subsequently the locations of all objects are sent to flat-mPL6 for detailed placement. Because of the random order of objects within each cluster, we believe there is still room for improvement even after the flat-mPL6 detailed placement. So we apply the detailed placer FastDP [6] to further refine the layout.

We normalize the results of flat-mPL6 with various pre-processing clustering to the results of flat-mPL6 *without* any pre-processing clustering. *For fair comparison, FastDP is applied to further refine the output layouts from the flat-mPL6 without pre-processing clustering*. We conduct four sets of experiments.

I. Clustering Targeting at Safe Cluster: We compare SC-G with FC and BC. FC's and BC's target γ is set the same as SC-G's. Table 2 shows that SC-G's HPWL is 2% worse than BC's and 1% better than FC's. For both clustering time and total time, SC-G is the *fastest*. Note that the cost C of some *unsafe* (i.e., $S_{max} > 0$) clusters may be better than some *safe* clusters. But unfortunately SC-G does not form any *unsafe* clusters. This makes SC-G's HPWL worse than BC's.

Table 2: Comparison with FirstChoice and BestChoice based on SC-G's clustering ratio (* comparison of scaled HPWL).

Circuit	Flat-mPL6		Clustering	Clustering Time (s)			Normalized HPWL to Flat-mPL6			Normalized Total Time to Flat-mPL6		
	HPWL ($\times 10e6$)	Time (s)	Ratio (γ)	FC	BC	SC-G	FC	BC	SC-G	FC	BC	SC-G
adaptec1	78.91	1197	0.80	1	2	8	1.00	1.00	1.00	1.61	1.47	1.24
adaptec2	90.71	1241	0.77	2	4	10	0.99	0.99	1.00	1.73	1.63	1.43
adaptec3	210.34	3923	0.71	6	23	33	1.00	0.99	0.99	1.38	1.51	1.17
adaptec4	188.39	3463	0.62	9	24	38	1.00	0.99	0.98	1.76	1.70	1.28
bigblue1	96.73	1424	0.77	2	4	11	0.99	0.99	1.00	1.61	1.46	1.60
bigblue2	146.98	3988	0.73	142	605	101	1.00	0.99	0.99	1.57	1.54	1.52
bigblue3	419.56	9486	0.58	35	123	91	0.91	0.88	0.90	1.01	1.00	1.04
bigblue4	812.89	10543	0.64	273	1529	287	1.00	0.99	0.99	1.47	1.41	1.34
adaptec5*	731.47	7892	0.68	60	263	95	0.87	0.74	0.81	1.04	1.20	1.14
newblue1*	109.85	17305	0.78	48	294	53	0.98	0.93	1.00	1.25	1.41	1.07
newblue2*	197.44	4396	0.68	19	62	57	1.00	0.99	0.99	1.04	0.95	1.06
newblue3*	320.63	10200	0.65	337	2393	228	0.94	0.96	0.95	1.29	1.65	1.67
newblue4*	438.99	7779	0.71	30	137	48	0.92	0.88	0.95	0.89	0.85	0.90
newblue5*	836.62	10124	0.66	363	1728	112	0.99	0.83	0.91	1.46	1.44	1.10
newblue6*	520.95	7575	0.74	572	3487	204	0.99	0.98	0.98	1.78	2.14	1.42
newblue7*	1076.36	19219	0.64	124	367	181	0.98	0.97	0.97	1.20	1.23	1.15
Average Normalized				**1.006**	**5.303**	**1**	**0.974**	**0.944**	**0.963**	**1.381**	**1.413**	**1.258**

Table 3: Comparison with FirstChoice, BestChoice and NetCluster based on NetCluster's clustering ratio (* comparison of scaled HPWL).

Circuit	Flat-mPL6		Clustering	Clustering Time (s)				Normalized HPWL to Flat-mPL6				Normalized Total Time to Flat-mPL6			
	HPWL ($\times 10e6$)	Time (s)	Ratio (γ)	FC	BC	NC	SC-R	FC	BC	NC	SC-R	FC	BC	NC	SC-R
adaptec1	78.91	1197	0.6381	1	3	69	20	1.00	1.00	1.01	0.99	0.92	0.91	1.15	1.04
adaptec2	90.71	1241	0.5764	2	6	63	30	1.01	1.00	1.00	0.99	1.22	1.11	1.11	1.28
adaptec3	210.34	3923	0.5677	7	24	62	98	1.02	0.99	0.99	0.99	1.15	1.09	1.00	1.04
adaptec4	188.39	3463	0.5382	8	26	58	86	1.01	1.00	0.98	0.98	1.19	1.13	1.07	1.15
bigblue1	96.73	1424	0.6128	2	5	66	23	0.99	0.98	0.98	0.98	1.19	1.08	1.21	1.13
bigblue2	146.98	3988	0.5977	195	814	64	181	1.02	1.00	0.99	0.99	0.98	1.11	0.89	0.86
bigblue3	419.56	9486	0.5074	36	144	53	163	0.92	0.87	0.89	0.88	0.81	0.81	0.74	0.76
bigblue4	812.89	10543	0.5617	315	1696	58	588	1.01	0.99	0.99	0.99	1.21	1.27	1.10	1.19
adaptec5*	731.47	7892	0.5569	81	335	60	284	0.87	0.73	0.79	0.69	0.98	0.98	0.90	0.92
newblue1*	109.85	17305	0.5674	90	472	62	125	0.93	0.90	1.03	0.86	0.82	0.88	0.77	0.83
newblue2*	197.44	4396	0.5886	22	65	65	92	1.02	1.00	1.10	1.00	0.74	0.81	0.69	0.77
newblue3*	320.63	10200	0.5462	427	2440	63	342	0.93	0.93	1.15	0.93	1.04	1.39	0.99	1.04
newblue4*	438.99	7779	0.6357	34	159	68	109	0.92	0.86	0.93	0.85	0.63	0.62	0.59	0.58
newblue5*	836.62	10124	0.5505	481	1860	58	214	0.92	0.81	0.84	0.79	1.08	1.07	0.95	1.13
newblue6*	520.95	7575	0.5836	868	4871	64	755	0.99	0.97	0.97	0.97	1.14	1.78	1.01	1.05
newblue7*	1076.36	19219	0.5634	142	423	60	519	0.99	0.97	0.99	0.97	0.89	0.87	0.89	0.98
Average Normalized				**0.545**	**2.475**	**0.813**	**1**	**0.971**	**0.937**	**0.978**	**0.928**	**1.000**	**1.056**	**0.940**	**0.985**

II. Clustering Targeting at NetCluster's Clustering Ratio: In this set of experiments, we compare SC-R with FC, BC and NC based on NC's γ. Since NC terminates when no more clusters can be formed, it cannot reach any γ as the users desire. For each circuit the target γ of other algorithms is set the same as NC's. As shown in Table 3, SC-R consistently generates the *best* HPWL for all 16 test cases, expect for one case (bigblue3) where SC-R is 1% worse than BC. On average SC-R generates 4%, 1% and 5% better HPWL than FC, BC and NC, respectively. In terms of clustering time, SC-R is $2.5\times$ faster than BC, while 45% and 19% slower than FC and NC, respectively. For the total time, SC-R is 1% and 7% faster than FC and BC, while 5% slower than NC.

III. Clustering Targeting at Various Clustering Ratios: We compare SC-R with FC and BC on five target clustering ratios $\gamma = 0.2, 0.3, 0.4, 0.5, 0.6$. In Table 4 the results are organized based on the circuits. We have two observations: (1) As γ goes lower, the clustering time increases but the total time generally decreases; (2) To improve the HPWL, for some circuits (e.g., adaptec5) it is good to cluster more objects. But for some circuits (e.g., newblue2) low γ degrades the HPWL. Fig. 3 shows the average normalized clustering time, HPWL and total time over all circuits for each γ. For clustering time, SC-R is faster than BC for all γ, except for $\gamma = 0.2$ where SC-R is 12% slower. For all γ, SC-R consistently produces the *best* HPWL compared with both FC and BC. Regarding the total time SC-R is consistently faster than BC. Even though SC-R is slower than FC on clustering time, SC-R's total time is very comparable with FC's, which means clusters produced by SC-R are preferred by the placer. Furthermore, considering the significant HPWL improvements over FC and the small percentage of clustering time over total time, we believe such slow down is acceptable.

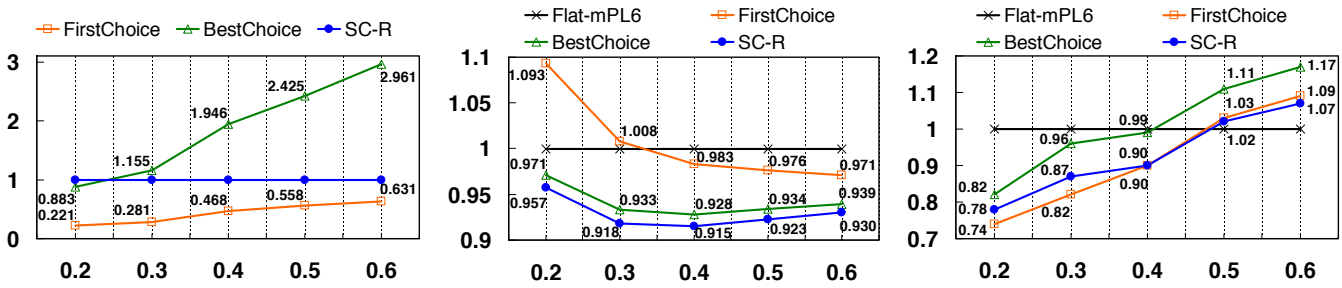
IV. Clustering Targeting at Best Placement Wirelength: Table 4 shows that various γ leads to various HPWL for each circuit. Here, we show that SC is able to automatically stop clustering, when the γ for the best HPWL is reached (see Table 5). Readers may compare Table 4 and Table 5 to verify this. To see how one-level clustering compares with multilevel clustering, we generate the results of original multilevel mPL6. *For fair comparison, FastDP is applied at the end of multilevel mPL6 (see the "mPL6+FastDP" columns in Table 5).* mPL6 has 4 levels of clustering and placement. The clustering time and final γ inside mPL6 are listed in Table 5. Even though SC on average generates 3% worse HWPL than mPL6, for almost half of the circuits SC's HPWL is even better than mPL6's. For most circuits, the HPWL generated by SC and mPL6 are very comparable. Regarding the total time, SC is significantly faster than mPL6 by 33%. Such results show that for some circuits one-level SC clustering generates better HPWL than multilevel BC clustering with substantial runtime speedup. From here we see prospective improvements if SC is applied into the multilevel placement framework.

5. CONCLUSION

In this paper, we have presented SafeChoice, a novel high-quality clustering algorithm. We aim at solving the fundamental problem — How to form safe clusters for placement. The clusters produced by SafeChoice are definitely essential for the placer to produce a good placement. Comprehensive experimental results show that SafeChoice is capable of producing the best clusters for the placer. To further improve and extend SafeChoice, our future work includes: (1) To develop the corresponding clustering considering object physical locations; (2) To integrate SafeChoice into a multilevel placement framework; (3) To derive the safe condition for more than two vertices. The source code of SafeChoice is publicly available at [19].

Table 4: Comparison with FirstChoice and BestChoice on target $\gamma = 0.2, 0.3, 0.4, 0.5, 0.6$ (* comparison of scaled HPWL).

Circuit	Flat-mPL6		Clustering	Clustering Time (s)			Normalized HPWL to Flat-mPL6			Normalized Total Time to Flat-mPL6		
	HPWL ($\times 10e6$)	Time (s)	Ratio (γ)	FC	BC	SC-R	FC	BC	SC-R	FC	BC	SC-R
adaptec1	78.91	1197	0.2	4	8	187	1.12	1.06	1.03	0.94	0.80	0.88
			0.3	3	6	121	1.05	1.02	1.00	0.90	0.80	0.92
			0.4	2	5	51	1.01	1.00	0.99	0.93	0.86	1.00
			0.5	2	4	35	1.00	1.00	0.99	0.95	0.92	0.91
			0.6	2	3	24	1.00	0.99	0.99	1.04	1.02	1.04
adaptec2	90.71	1241	0.2	8	16	238	1.08	1.02	1.00	1.03	0.82	1.01
			0.3	5	12	144	1.05	0.99	0.98	1.35	1.20	1.22
			0.4	4	9	59	1.03	1.01	0.98	1.38	1.19	1.23
			0.5	3	7	39	1.01	1.00	0.98	1.43	1.19	1.28
			0.6	3	6	26	1.00	0.99	0.98	1.45	1.24	1.20
adaptec3	210.34	3923	0.2	19	46	572	1.15	1.02	1.02	0.77	0.70	0.76
			0.3	14	38	390	1.08	1.00	0.99	0.79	0.72	0.81
			0.4	11	32	162	1.04	1.00	0.99	0.94	0.74	0.73
			0.5	9	26	114	1.04	1.00	0.98	1.28	1.16	1.17
			0.6	7	23	80	1.01	1.00	0.98	1.35	1.21	1.23
adaptec4	188.39	3463	0.2	16	49	403	1.08	0.99	0.99	0.81	0.70	0.79
			0.3	13	42	276	1.04	0.98	0.98	0.82	0.74	0.77
			0.4	11	35	130	1.02	0.99	0.98	0.83	0.75	0.85
			0.5	9	30	89	1.01	0.99	0.98	1.28	1.26	1.18
			0.6	7	22	59	1.00	0.99	0.99	1.16	1.29	1.18
bigblue1	96.73	1424	0.2	8	15	297	1.05	1.01	1.02	0.86	0.68	0.98
			0.3	5	12	179	1.02	0.98	0.99	0.85	0.95	0.98
			0.4	4	9	66	1.01	0.98	0.98	0.91	0.86	0.91
			0.5	3	7	39	1.00	0.97	0.98	1.01	0.85	0.89
			0.6	2	6	23	1.00	0.98	0.98	1.09	1.17	1.18
bigblue2	146.98	3988	0.2	395	1749	1162	1.15	1.07	1.05	0.84	1.05	0.92
			0.3	344	1516	667	1.07	1.01	1.01	0.86	1.16	0.91
			0.4	302	1295	359	1.04	1.00	0.99	0.88	1.14	0.90
			0.5	244	1005	226	1.03	0.99	0.99	0.95	1.18	0.92
			0.6	194	796	159	1.01	1.00	0.99	1.09	1.11	0.99
bigblue3	419.56	9486	0.2	69	264	800	0.92	0.82	0.85	0.52	0.49	0.57
			0.3	55	221	492	0.92	0.87	0.83	0.78	0.76	0.75
			0.4	46	174	241	0.92	0.84	0.85	0.90	0.81	0.82
			0.5	36	146	158	0.93	0.88	0.88	0.93	0.95	0.91
			0.6	30	116	89	0.91	0.88	0.89	1.00	1.01	0.91
bigblue4	812.89	10543	0.2	633	2907	3262	1.12	1.01	1.02	1.06	1.17	1.19
			0.3	534	2576	2220	1.06	1.00	0.99	1.19	1.44	1.24
			0.4	451	2169	1145	1.03	0.99	0.99	1.16	1.45	1.14
			0.5	368	1819	733	1.01	0.99	0.99	1.24	1.35	1.23
			0.6	288	1453	434	1.01	0.99	0.99	1.27	1.38	1.22
adaptec5*	731.47	7892	0.2	165	569	1424	0.77	0.63	0.62	0.52	0.56	0.66
			0.3	139	503	984	0.83	0.66	0.63	0.52	0.52	0.62
			0.4	114	419	456	0.84	0.68	0.65	0.61	0.58	0.59
			0.5	93	358	324	0.86	0.72	0.69	1.07	1.29	1.07
			0.6	73	311	204	0.88	0.73	0.70	1.23	1.32	1.15
newblue1*	109.85	17305	0.2	169	806	781	0.91	0.88	0.81	0.13	0.24	0.23
			0.3	149	718	527	0.91	0.86	0.80	0.23	0.49	0.41
			0.4	127	630	226	0.91	0.87	0.81	0.30	0.57	0.39
			0.5	104	538	141	0.93	0.89	0.84	0.91	1.30	0.96
			0.6	84	434	93	0.94	0.90	0.86	1.05	1.31	1.04
newblue2*	197.44	4396	0.2	43	200	415	2.16	1.58	1.44	0.74	0.69	0.81
			0.3	37	181	278	1.29	1.11	1.11	0.86	0.88	0.76
			0.4	32	164	155	1.07	1.03	1.03	0.82	0.89	0.86
			0.5	26	128	116	1.02	1.01	1.01	1.04	0.88	0.97
			0.6	21	65	84	1.01	1.00	1.00	0.90	0.89	0.90
newblue3*	320.63	10200	0.2	931	3789	1010	0.98	0.89	0.89	0.50	0.76	0.50
			0.3	783	3480	692	0.93	0.89	0.90	0.90	1.60	1.00
			0.4	630	3041	407	0.92	0.90	0.91	0.95	1.56	1.14
			0.5	487	2546	326	0.92	0.93	0.93	1.00	1.34	1.15
			0.6	362	2299	256	0.94	0.95	0.95	1.05	1.41	1.23
newblue4*	438.99	7779	0.2	77	334	981	0.94	0.88	0.81	0.46	0.57	0.49
			0.3	66	302	643	0.91	0.88	0.80	0.50	0.64	0.60
			0.4	55	267	275	0.91	0.86	0.81	0.59	0.72	0.62
			0.5	46	221	188	0.92	0.86	0.81	0.53	0.61	0.57
			0.6	37	168	114	0.93	0.85	0.83	0.62	0.59	0.63
newblue5*	836.62	10124	0.2	1093	3948	1483	0.94	0.70	0.78	0.64	1.02	0.62
			0.3	877	2863	935	0.95	0.73	0.74	0.71	0.95	0.69
			0.4	693	2124	392	0.96	0.77	0.77	1.09	1.10	1.01
			0.5	532	1903	237	0.94	0.78	0.77	1.04	1.10	0.98
			0.6	399	1713	155	0.92	0.82	0.80	1.04	1.10	0.94
newblue6*	520.95	7575	0.2	1941	8229	4058	1.05	0.99	0.97	1.16	1.95	1.19
			0.3	1641	7415	2793	1.01	0.97	0.96	1.11	1.77	1.31
			0.4	1343	6558	1378	1.00	0.97	0.96	1.27	1.88	1.28
			0.5	1082	5391	890	0.99	0.97	0.97	1.14	1.64	1.07
			0.6	824	4535	639	0.99	0.98	0.97	1.17	1.66	1.16
newblue7*	1076.36	19219	0.2	290	948	2704	1.07	0.99	1.00	0.90	0.98	0.81
			0.3	238	738	1774	1.02	0.97	0.97	0.78	0.80	0.93
			0.4	197	605	891	1.00	0.97	0.97	0.79	0.74	0.95
			0.5	159	472	596	0.99	0.97	0.97	0.76	0.72	1.00
			0.6	126	380	422	0.98	0.97	0.97	0.93	1.00	1.03



(a) Normalized clustering time to SC-R (b) Normalized HPWL to flat-mPL6 (c) Normalized total time to flat-mPL6

Figure 3: Average normalized clustering time, HPWL and total time over all circuits for target $\gamma = 0.2, 0.3, 0.4, 0.5, 0.6$.

Table 5: Comparison with original multilevel mPL6 (* comparison of scaled HPWL).

Circuit	HPWL ($\times 10e6$)			Total Time (s)			SC Clustering Info.		BC Clustering Info. inside mPL6		
	Flat-mPL6	SC	mPL6+FastDP	Flat-mPL6	SC	mPL6+FastDP	Time (s)	γ	Time (s)	Final γ	# of levels
adaptec1	78.91	78.51	76.47	1197	1238	1807	76	0.33	29	0.006	4
adaptec2	90.71	**88.51**	89.19	1241	2064	2032	73	0.36	47	0.006	4
adaptec3	210.34	207.27	206.00	3923	3732	6187	228	0.33	87	0.006	4
adaptec4	188.39	**184.33**	187.51	3463	3227	5687	208	0.31	67	0.007	4
bigblue1	96.73	95.31	95.14	1424	1319	2208	109	0.32	42	0.008	4
bigblue2	146.98	**146.07**	146.57	3988	4183	5992	458	0.36	81	0.045	4
bigblue3	419.56	357.56	331.70	9486	10516	8842	420	0.30	131	0.005	4
bigblue4	812.89	**803.43**	806.83	10543	15460	19457	1622	0.33	468	0.008	4
adaptec5*	731.47	461.99	429.97	7892	5919	10796	697	0.32	149	0.005	4
newblue1*	109.85	88.10	64.72	17305	7490	2567	368	0.31	44	0.005	4
newblue2*	197.44	**198.35**	198.90	4396	6303	7141	91	0.58	61	0.007	4
newblue3*	320.63	287.76	283.25	10200	14986	9644	683	0.30	66	0.029	4
newblue4*	438.99	351.02	301.89	7779	6053	9481	421	0.33	93	0.010	4
newblue5*	836.62	624.26	526.98	10124	8405	16220	625	0.34	251	0.008	4
newblue6*	520.95	**498.44**	516.43	7575	11081	13566	2059	0.33	255	0.009	4
newblue7*	1076.36	**1042.97**	1070.08	19219	21049	32561	1159	0.34	278	0.014	4
Normalized	**1**	**0.910**	**0.879**	**1**	**1.086**	**1.412**					

Acknowledgment

The authors would like to thank Guojie Luo from UCLA CAD group and Logan Rakai from University of Calgary for the help with mPL6 and NetCluster, respectively.

6. REFERENCES

[1] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *Proc. ICCAD*, pages 187–192, 2006.

[2] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov. Min-cut floorplacement. *IEEE Trans. on Computer-Aided Design*, 25(7):1313–1326, July 2006.

[3] B. Hu and M. Marek-Sadowska. Multilevel fixed-point-addition-based vlsi placement. *IEEE Trans. on Computer-Aided Design*, 24(8):1188–1203, August 2005.

[4] A. B. Kahng and Q. Wang. A faster implementation of APlace. In *Proc. ISPD*, pages 218–220, 2006.

[5] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie. mPL6: Enhanced multilevel mixed-sized placement. In *Proc. ISPD*, pages 212–214, 2006.

[6] N. Viswanathan, M. Pan, and C. Chu. FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Proc. ASP-DAC*, pages 135–140, 2007.

[7] N. Viswanathan, G.-J. Nam, C. Alpert, P. Villarrubia, H. Ren, and C. Chu. RQL: Global placement via relaxed quadratic spreading and linearization. In *Proc. DAC*, pages 453–458, 2007.

[8] C. J. Alpert and A. B. Kahng. Recent developments in netlist partitioning: A survey. *Integration, the VLSI Journal*, 19(1-2):1–81, August 1995.

[9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: application in VLSI domain. In *Proc. DAC*, pages 526–529, 1997.

[10] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proc. DAC*, pages 343–348, 1999.

[11] C. J. Alpert, J.-H. Huang, and A. B. Kahng. Multilevel k-way hypergraph partitioning. In *Proc. DAC*, pages 530–533, 1997.

[12] T. Chan, J. Cong, and K. Sze. Multilevel generalized force-directed method for circuit placement. In *Proc. ISPD*, pages 185–192, 2005.

[13] J. Cong and S. K. Lim. Edge separability-based circuit clustering with application to multilevel circuit partitioning. *IEEE Trans. on Computer-Aided Design*, 23(3):346–357, March 2004.

[14] B. Hu and M. Marek-Sadowska. Fine granularity clustering-based placement. *IEEE Trans. on Computer-Aided Design*, 23(4):527–536, April 2004.

[15] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng. A fast hierarchical quadratic placement algorithm. *IEEE Trans. on Computer-Aided Design*, 25(4):678–691, April 2006.

[16] J. Li, L. Behjat, and J. Huang. An effective clustering algorithm for mixed-size placement. In *Proc. ISPD*, pages 111–118, 2007.

[17] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 placement contest and benchmarks suite. In *Proc. ISPD*, pages 216–220, 2005.

[18] G.-J. Nam. ISPD 2006 placement contest: Benchmark suite and results. In *Proc. ISPD*, pages 167–167, 2006.

[19] SafeChoice source code. `http://www.public.iastate.edu/~zijunyan/`.