SafeChoice: A Novel Approach to Hypergraph Clustering for Wirelength-Driven Placement

Jackey Z. Yan, Chris Chu, and Wai-Kei Mak

***Abstract*—This paper presents a completely new approach to the problem of hypergraph clustering for wirelength-driven placement. The novel algorithm we propose is called SafeChoice (SC). Different from all previous approaches, SC is proposed based on a fundamental theorem, *safe condition* which guarantees that clustering would not degrade the placement wirelength. To mathematically derive such a theorem, we first introduce the concept of *safe clustering*, i.e., do clustering without degrading the placement quality. To efficiently check the safe condition for pair-wise clustering, we propose a technique called *selective enumeration*. SafeChoice maintains a global priority queue based on the safeness and area of potential clusters. Using a simple heuristic, it automatically stops clustering when generating more clusters would degrade the placement wirelength. Moreover, we extend SafeChoice to do clustering while considering the object physical locations, i.e., *physical clustering*. Finally, we apply SafeChoice into a two-phase placement framework and propose a high-quality analytical placement algorithm called SCPlace. Comprehensive experimental results show that the clusters produced by SC consistently help the placer to achieve the best wirelength among all other clustering algorithms, and SCPlace generates the best half-perimeter wirelength compared with all other state-of-the-art placers.**

***Index Terms*—Hypergraph clustering, physical design, VLSI placement.**

I. Introduction

FOR MODERN very large scale integration (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]–[8].

Manuscript received April 21, 2010; revised August 11, 2010 and November 13, 2010; accepted January 19, 2011. Date of current version June 17, 2011. This work was supported in part by the IBM Faculty Award, NSF, under Grant CCF-0540998, and NSC, under Grant NSC 99-2220-E-007-007. This paper was recommended by Associate Editor Y.-W. Chang.

J. Z. Yan is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010 USA, and also with the Placement Technology Group, Cadence Design Systems, Inc., San Jose, CA 95134 USA (e-mail: zijunyan@iastate.edu).

C. Chu is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50010 USA (e-mail: cnchu@iastate.edu).

W.-K. Mak is with the Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan (e-mail: wkmak@cs.nthu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2011.2114950

A. Previous Work

Clustering is a traditional problem in VLSI computer-aided design (CAD) area. The clustering algorithms proposed long time ago were described in [9]. In the last several years, various new algorithms were proposed to continue improving the clustering quality. In [10], 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 [11] 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.* [12] and Chan *et al.* [13] 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.* [14] 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.* [15] 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.* [16] proposed BestChoice (BC) clustering which has been widely used in the top-of-the-line placers APlace [4], mPL6 [5], FastPlace3 [6], RQL [7], and FLOP [8]. 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 tradeoff between the clustering runtime and quality.

Fig. 1. Example of indirect connections between objects a and b.

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.* [17] 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 [18]. 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 direct connectivity among the objects, so that the most highly directly 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 a constraint is enforced improperly, i.e., clustering objects that should not be clustered, the placement solution would be jeopardized. It has not been proved that clustering highly directly connected objects can definitely minimize the placement wirelength. Even though it makes some sense intuitively to cluster such objects, we believe it is not sufficient to just consider the *direct* connections. We also need to take the *indirect* connections into account. For example in Fig. 1, two objects a and b are connected by a two-pin net. At the same time, they are indirectly connected by two two-pin nets via object c. Such indirect connections intend to pull a and b toward each other. But they have been ignored in all previous work. As a result, it is very likely that previous algorithms mislead the placers to a low-quality solution.

In order to form the best clusters for placement, we need to solve the fundamental problem of clustering for placement: *how can we do clustering so that it can be guaranteed that clustering would not degrade the placement quality?*

B. Our Contributions

This paper presents a completely new approach to the problem of hypergraph clustering for wirelength-driven placement. We propose a novel clustering algorithm called SafeChoice (SC).[1] 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 seven main contributions.

1) *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.
2) *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.
3) *Selective enumeration*: to check the safe condition for pair-wise clustering, we propose *selective enumeration*. With such a method, we can efficiently find out the safe clusters in a circuit.
4) *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.
5) *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.
6) *Physical SafeChoice*: we extend SafeChoice to do clustering if the physical locations of some objects are given. In this way, SafeChoice can make use of such location information, e.g., an initial placement or fixed I/O object locations, and thus produces even better clusters.
7) *SCPlace*: to demonstrate the effectiveness of physical SafeChoice, we propose a simple and high-quality two-phase placement algorithm called SCPlace. SCPlace is simple in the sense that it has only one clustering level and two placement phases. But, it produces significantly better results than all other state-of-the-art placement algorithms.

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 help the placer to generate the best wirelength. Compared with the state-of-the-art placement algorithms, SCPlace is able to generate the best half-perimeter wirelength (HPWL).

The rest of this paper is organized as follows. Section II describes the safe clustering. Section III introduces the algorithm of SafeChoice. Section IV presents the physical SafeChoice. Section V introduces the algorithm of SCPlace. Experimental results are presented in Section VI. Finally, this paper ends with a conclusion and the direction of future work.

II. 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, 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 legalized placements

[1] A preliminary version of SafeChoice was presented in [19].

of the vertices in V. The wirelength is measured by weighted HPWL.

A. 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.

Definition 2 is established based an assumption that the area of every vertex in V_c is zero, so that we can move them in a legalized placement and ignore the overlap issue. In other words, we only consider clustering zero-area objects. Definition 2 shows that when safe clustering is performed on any legalized (zero-overlap) placement p, it does not increase the total wirelength of p. But, the clustering algorithm is not a placement algorithm and thus does not specify how any overlap incurred by the clustering is to be removed. Therefore, we assume that the clusters are small enough, i.e., zero area, compared to the total area of place-able objects in the clustered netlist, such that any wirelength increase incurred by any displacement needed to remove overlap incurred by the clustering does not exceed the wirelength decrease induced by the clustering. This assumption is reasonable, even though there is no zero-area object in the real circuits. This is because for a typical clustering ratio,[2] the size of each cluster is always much smaller than the total area of objects in the circuit. Note that, however, such assumption may not be applicable when some complex floorplan geometry is presented, e.g., when a big cluster is placed in a narrow channel between two big fixed objects. In this case, the displacement needed to remove overlap incurred by the clustering may be quite big.

As you can see, 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.

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 $p \in P$, first we move those vertices horizontally to the same x coordinate. Second, 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.

In the remaining part of Section II, we consider only x direction and horizontally safe clustering. Analogically, the theoretical proof and mathematical derivation for y direction and vertical safe clustering can be done in a similar way.

B. 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. First, 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., $V_c = \{a, b\}$), and a is on the left of b.

Definition 4: wirelength gradient function. Given a placement $p \in P$ and a hyperedge $e \in E$, we define

$$\Delta_a(p, e): \text{gradient function of wirelength of } e \text{ if } a \text{ is moving toward } b;$$
$$\Delta_b(p, e): \text{gradient function of wirelength of } e \text{ if } b \text{ is moving toward } a.$$

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 toward 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 toward 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 toward b would not increase the total wirelength; if $\mathcal{F}_{ab}(p) > 0$, then moving either a or b toward 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 toward 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 toward a would not increase the total wirelength. Again, after such movement, we get another placement p^2

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

with total wirelength l^2, where $l^1 \geq l^2$. We keep moving either a or b toward 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. ■

C. 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

$$\begin{aligned} &\forall e \in E_a \quad \Delta_a(p, e) = \Delta_a(p', e) \\ &\forall e \in E_b \quad \Delta_b(p, e) = \Delta_b(p', e). \end{aligned}$$

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 $k_2 \geq x_b$; and 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[3] possible values of $\Delta_a(p, e)$.

1) $\Delta_a(p, e) = w_e$.
   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, $\Delta_a(p', e) = w_e = \Delta_a(p, e)$.
2) $\Delta_a(p, e) = 0$.
   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

$$\begin{aligned} &\forall e \in E_a \quad \Delta_a(p, e) = \Delta_a(p', e) \\ &\forall e \in E_b \quad \Delta_b(p, e) = \Delta_b(p', e). \end{aligned}$$

Thus, $\mathcal{F}_{ab}(p) = \mathcal{F}_{ab}(p')$. Analogically, the cases for (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 ($x_v \geq x_b$); and 3) v is between a and b ($x_a < x_v < x_b$).

Based on Lemmas 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

$$\forall e \in E_v \quad \Delta_a(p, e) \leq \Delta_a(p', e). \tag{1}$$

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

$$\forall e \in E_v \quad \Delta_b(p, e) = \Delta_b(p', e). \tag{2}$$

Based on (1) and (2), we have

$$\begin{aligned} &\forall e \in E_a \quad \Delta_a(p, e) \leq \Delta_a(p', e) \\ &\forall e \in E_b \quad \Delta_b(p, e) = \Delta_b(p', e). \end{aligned}$$

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. ■

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. 2(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

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

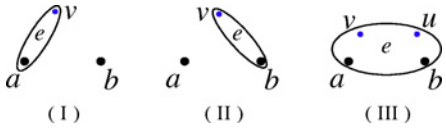


Fig. 2. Simple examples of vertices that can be fixed.

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. Let N_a denote the set of vertices sharing at least one hyperedge with vertex a, and $\overline{N}_a = V - N_a$. Similarly, we can define N_b and $\overline{N}_b$.

1) $V^I = N_a \cap \overline{N}_b$ [e.g., in Fig. 2(I) vertex $v \in V^I$].
2) $V^{II} = \overline{N}_a \cap N_b$ [e.g., in Fig. 2(II) vertex $v \in V^{II}$].
3) $V^{III} = \{v | v \in V_{ab}$ s.t. $(E_v \cap (E_a \cup E_b)) \subset (E_a \cap E_b)\}$ [e.g., in Fig. 2(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 $E_{v \cap ab} = E_v \cap (E_a \cup E_b)$.

1) In placement p, $\forall e \in E_{v \cap ab}$, we consider two cases.
   a) $\exists$ 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 $\Delta_a(p, e) = 0$.
   b) $\nexists$ 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, $\forall e \in E_{v \cap ab}$, $\Delta_a(p, e) \geq 0$.
2) In placement p', $\forall e \in E_{v \cap ab}$, we consider two cases.
   a) $\exists$ vertex $c \in e (c \neq a, c \neq v)$, 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 $\Delta_a(p', e) = 0$.
   b) $\nexists$ 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

$$\begin{aligned} &\forall e \in E_a \quad \Delta_a(p, e) \geq \Delta_a(p', e) \\ &\forall e \in E_b \quad \Delta_b(p, e) = \Delta_b(p', e). \end{aligned}$$

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.

1) *v moved to the right of b, u did not move.*
   In placement p', $\forall e \in E_{v-u}$ we consider two cases.
   a) $\exists$ 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 $\Delta_a(p', e) = 0$, $\Delta_b(p', e) = 0$.
   b) $\nexists$ 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 $\Delta_a(p', e) = -w_e$, $\Delta_b(p', e) = 0$.
2) *u moved to the left of a, v did not move.*
   In placement p', $\forall e \in E_{v-u}$ we consider two cases.
   a) $\exists$ 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 $\Delta_a(p', e) = 0$, $\Delta_b(p', e) = 0$.
   b) $\nexists$ 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 $\Delta_a(p', e) = 0$, $\Delta_b(p', e) = -w_e$.
3) *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

$$\begin{aligned} &\forall e \in E_a \quad \Delta_a(p, e) \geq \Delta_a(p', e) \\ &\forall e \in E_b \quad \Delta_b(p, e) \geq \Delta_b(p', e). \end{aligned}$$

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. 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

$$s_{max} = max(s_1, s_2, ..., s_L). \tag{3}$$

Based on Theorem 1, if $s_{max} \leq 0$, then it is safe to cluster a and b. The flow of selective enumeration is shown in Fig. 3.

The more placements we enumerate (i.e., the bigger L is), the slower the algorithm runs. To limit the runtime, at most 2^{10} placements are enumerated by default. If $|V_{ab}| - \alpha > 10$,

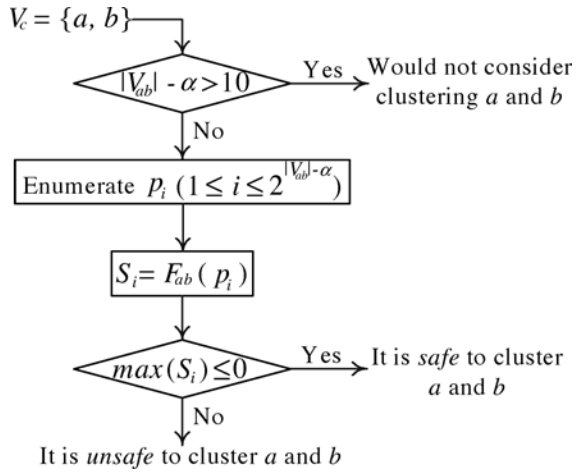


Fig. 3. Flow of selective enumeration.

TABLE I
PROFILE OF SELECTIVE ENUMERATION FOR EACH CIRCUIT

Circuit	Number of Cases $\lvert V_{ab}\rvert - \alpha <= 10$ (%)	Number of Cases $\lvert V_{ab}\rvert - \alpha > 10$ (%)	Maximum $\lvert V_{ab}\rvert - \alpha$
adaptec1	3 079 149 (94%)	205 937 (6%)	78
adaptec2	2 834 333 (92%)	252 967 (8%)	110
adaptec3	5 568 171 (90%)	591 043 (10%)	191
adaptec4	6 362 256 (93%)	448 166 (7%)	110
bigblue1	3 677 691 (94%)	218 500 (6%)	154
bigblue2	6 941 283 (95%)	351 816 (5%)	1582
bigblue3	12 136 725 (95%)	592 998 (5%)	205
bigblue4	22 954 456 (92%)	2 104 452 (8%)	1102
adaptec5	9 935 184 (90%)	1 058 753 (10%)	278
newblue1	4 610 210 (97%)	159 220 (3%)	608
newblue2	4 058 091 (98%)	73 838 (2%)	144
newblue3	6 758 853 (93%)	489 111 (7%)	1016
newblue4	9 215 574 (95%)	530 334 (5%)	331
newblue5	12 046 236 (94%)	744 213 (6%)	372
newblue6	15 996 670 (91%)	1 527 619 (9%)	1316
newblue7	26 414 433 (94%)	1 700 947 (6%)	1151

we simply would not consider clustering a and b, and consequently we may lose some potential safe clusters. Table I shows the number of cases where $|V_{ab}| - \alpha <= 10$ and $|V_{ab}| - \alpha > 10$, and also the maximum value of $|V_{ab}| - \alpha$ for each ISPD 05/06 circuit. As you can see, in practice we have $|V_{ab}| - \alpha \le 10$ for most of the pairs (i.e., more than 90% of the pairs). Even if the unconsidered pairs are all safe, we would only lose a very small portion of safe clusters.

III. ALGORITHM OF SAFECHOICE

In the previous section, we have described a practical method of checking the safe condition for pair-wise clustering. As shown in Definition 3, the safe condition has to be checked both horizontally and vertically. However, 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, which means it is sufficient to check the safe condition only in x direction. Therefore, in this section we ignore the fixed objects by treating them the same as movable ones, apply selective enumeration in a PQ-based algorithm flow and propose SafeChoice algorithm. To satisfy various clustering objectives, we present three operation modes for SafeChoice.

TABLE II
DIFFERENCES OF 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

A. Priority-Queue Based Framework

Previous work [12], [13] showed 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. Moreover, because the concept of safe clustering in Definition 2 is defined based on the assumption of clustering zero-area objects, to apply such concept on real circuits we need relax this assumption and cluster small objects.[4] Thus, 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:

$$C(a, b) = S^* + \theta \times \frac{A_a + A_b}{\overline{A}_s} \tag{4}$$

where θ is the weight between the safeness and area (based on the experiments $\theta = 4$ 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 III-B).

In SafeChoice we maintain a global PQ similar to that in [16]. But we rank each pair of objects based on the cost obtained by (4). If two pairs have the same cost, we just randomly determine the priority between them. 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 (based on the experiments $\beta = 7$ 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 III-B.

B. Operation Modes of SafeChoice

Given a circuit, some algorithms (e.g., FC and BC) can reach any clustering ratio γ, 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 best γ is reached. By default SafeChoice automatically stops clustering

[4]After relaxing the assumption, moving the small objects may create small overlap. But as the objects are small, the result placement can be legalized by slightly shifting objects around and the impact to HPWL should be minimal.

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 II).

1) *Safety guarantee mode (SC-G):* SC-G aims at producing the completely safe clusters. Under this mode, $S^* = s_{max}$ in (4). 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.
2) *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 tradeoff 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:

$$\overline{s} = \frac{\sum_{i=1}^{L} s_i}{L}. \tag{5}$$

Basically, for a pair of objects a and b (5) calculates the average score $\overline{s}$ over the L enumerated placements. Under SC-R mode, $S^* = \overline{s}$ in (4). Iteratively, SC-R clusters the pair of objects at the top of the PQ until the target γ is reached.
3) *Smart mode (SC) (default mode):* using a simple heuristic, the smart mode stops the clustering process when a typical placer achieves the best placement wirelength. None of previous clustering algorithms has such feature. For different circuits, the γ for the best placement wirelength may be different. In SC, we set a threshold cost C_t, and use the same cost function as in SC-R. During the clustering process, SC would not terminate until the cost reaches C_t. Based on the experimental results, we set $C_t = 21$ by default. With this simple heuristic, SC is able to automatically stop when generating more clusters starts to degrade the placement wirelength.

IV. Physical SafeChoice

In this section, we extend SafeChoice to do clustering while considering the object physical locations, i.e., *physical clustering*.

Compared with non-physical clustering algorithms, physical clustering is to do clustering based on both the netlist connectivity information and the object physical locations. Such physical locations can be obtained from an initial placement or existing fixed objects. It has been shown in [3], [6], [7], and [20] that the physical clustering can significantly improve the clustering quality. For SafeChoice, it is very natural to be extended to physical clustering. This is because SafeChoice applies selective enumeration to enumerate different placements. If an initial placement is given, many more placements can be ignored in the enumeration. This simplifies the enumeration process by pruning away the placements that would not be possibly generated.

In the following subsections, we first introduce the safe condition for physical SafeChoice. After that, we present how to further reduce the enumeration size based on the given physical information. Finally, we show the corresponding changes of cost functions in different modes of physical SafeChoice.

A. Safe Condition for Physical SafeChoice

Because in physical SafeChoice the fixed objects are taken into account, horizontally safe may not always imply vertically safe. Therefore, we need to consider the safe condition for both x and y directions in physical SafeChoice.

Definitions 4 and 5 are defined for x direction. But they can be easily extended to y direction. Based on Theorem 1, we present the safe condition for physical SafeChoice as follows.

Theorem 2: safe condition for $V_c = \{a, b\}$ in physical SafeChoice. It is safe to cluster a and b if $\forall p \in P, \mathcal{F}^x_{ab}(p) \leq 0$, and $\mathcal{F}^y_{ab}(p) \leq 0$.

In Theorem 2, $\mathcal{F}^x_{ab}(p)$ and $\mathcal{F}^y_{ab}(p)$ are the total wirelength gradient functions for x and y directions, respectively. In Section II-A we have proved that if vertices in V_c are both horizontally and vertically safe for clustering, then it is safe to cluster them. Therefore, Theorem 2 can be proved similarly to Theorem 1.

B. Enumeration Size Reduction Based on Physical Location

In this subsection, we use the physical location information to cut down the placement enumeration size. In this following discussion, we assume that such physical locations are derived from an initial placement.

First of all, for each object we define a square-shape region to differentiate between the "long" and "short" distances (in one dimension) of two objects. The center of each square-shape region is the corresponding object location in the initial placement. The insight is that we assume in the final placement the objects would not be placed outside of their regions. So intuitively, the better the initial placement is, the less displacements of object locations between the initial and final placements we have, and thus the smaller such regions are. Let t denote the side length of each square-shape region. $\forall v, u \in V$, D^x_{vu} denotes the distance of two vertices v and u in x direction. Therefore, based on the square-shape region, we can derive the following three scenarios in x direction to cut down the enumeration size (the scenarios for y direction can be derived similarly).

1) If $D^x_{ab} > t$, then we would not consider to cluster a and b [see Fig. 4(a)].
2) If $x_c \leq x_a$ and $D^x_{cb} > t$, then in selective enumeration we fix c on the left of a [see Fig. 4(b)].
3) If $x_c \geq x_b$ and $D^x_{ca} > t$, then in selective enumeration we fix c on the right of b [see Fig. 4(c)].

Scenario 1 is used as a filter to prune away the pairs of objects that would not be clustered due to the "long" distance between them. Scenarios 2 and 3 are applied within selective enumeration to identify the subsets of vertices that can be fixed. In Scenario 2 the reason why we consider "$D^x_{cb} > t$," instead of "$D^x_{ca} > t$," is that as long as c is unlikely to be on the

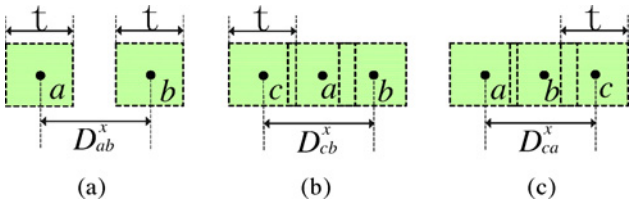


Fig. 4. Examples of three scenarios with square-shape region. (a) Scenario 1. (b) Scenario 2. (c) Scenario 3.

TABLE III
S^* FOR THE THREE MODES IN PHYSICAL SAFECHOICE

Mode	S^*
SC-G	$max(s^x_{max}, s^y_{max})$
SC-R	$(\bar{s}^x + \bar{s}^y)/2$
SC	$(\bar{s}^x + \bar{s}^y)/2$

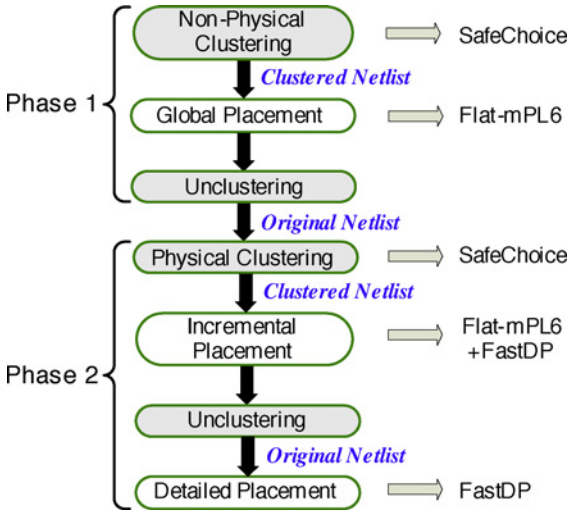


Fig. 5. Simple two-phase placement flow in SCPlace.

right of b, it will definitely be on the left of a. This is because as shown in Lemma 3, there are only two possible locations to enumerate for c, i.e., on the left of a and on the right of b. The same reason applies for "$D^x_{ca} > t$" in Scenario 3.

Ideally, for different objects t should be different. However, finding such a t accurately for each object is still an open problem. After all, we just need a simple estimation on the displacement that is roughly determined by the quality of the placer. Therefore, by default we set $t = 15 \times h_{row}$ based on the experiments and our experience on placement, where h_{row} is the placement row height.

C. Cost Function for Physical SafeChoice

In physical SafeChoice, we employ almost the same cost function as (4). However, the S^* term in (4) is defined based on one dimension, i.e., x direction. So we need to extend it to two dimensions. The S^* for the three operation modes in physical SafeChoice are listed in Table III, where s^x_{max} and $\bar{s}^x$ denote the s_{max} in (3) and $\bar{s}$ in (5) for x direction, respectively, similarly for s^y_{max} and $\bar{s}^y$.

V. SAFECHOICE-BASED TWO-PHASE PLACEMENT

In this section, we first propose a simple two-phase placement flow. Then based on this new algorithm flow, we present SCPlace, a high-quality analytical placement algorithm.

The state-of-the-art placement algorithms [1]–[7] all adopt a multilevel framework to cope with the ever-increasing complexities of modern VLSI placement. At each hierarchical level of the coarsening phase, the placers first do clustering on the netlist passed from previous level, and then do placement of the clustered netlist. Such a multilevel clustering-placement process does not stop, until the original circuit is reduced to a reasonably smaller size. Subsequently, a corresponding multilevel unclustering-placement process is applied at the uncoarsening phase. Typically, modern placers contain at least four levels of clustering and placement.

Different from previous work, we propose a simple two-phase placement flow shown in Fig. 5. It is simple in the sense that this flow contains only one level of clustering and two phases of placement. The goal of the first phase is to generate an initial placement, and to provide the physical location information for the physical clustering in the next phase. Then at the second phase, we apply physical clustering on the original netlist rather than the clustered netlist from previous clustering. This is the key difference between our physical clustering scheme and the ones in [3], [6], [7], and [20]. The reason for doing this is that non-physical clustering may produce some low-quality clusters due to the lack of physical information. In order to correct such mistake in non-physical clustering, in physical clustering we form the clusters from the original netlist. The location of each such cluster is calculated as the average location of all objects in that cluster. As a result, after physical clustering we have an updated location for each object in the clustered netlist. Subsequently, we start an incremental placer based on such physical information to do both global and detailed placement on the clustered netlist. Finally, after unclustering we use the detailed placement algorithm to refine the layout.

Based on this simple two-phase placement flow, we implement a high-quality analytical placement algorithm, SCPlace (see Fig. 5). We use SafeChoice as the clustering algorithm inside SCPlace. As mentioned above, non-physical clustering may produce some low-quality clusters due to the lack of physical information, so we want SafeChoice to generate less such clusters. Therefore, we set $C_t = 16$ instead of $C_t = 21$. For the physical SafeChoice in the second phase, we use the default $C_t = 21$. mPL6 [5] is applied as the placement engine. However, mPL6 is based on a multilevel framework, and uses BC as its internal clustering algorithm. Without turning off BC inside mPL6, we cannot demonstrate the effectiveness of SafeChoice, because the internal clustering process will produce some noise to the results. Therefore, we turn off the BC clustering inside mPL6 by adding "-cluster_ratio 1" to the command line, so that mPL6 performs only one-level placement without any clustering inside, i.e., flat-mPL6.[5] The detailed placer FastDP [6] is used as the additional detailed placement algorithm in SCPlace.

[5]As far as we know, mPL6 is the only placer that can turn off the internal clustering without modifying the source code.

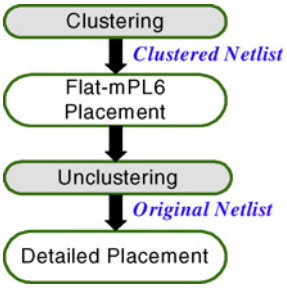


Fig. 6. Experimental flow for clustering algorithm.

VI. Experimental Results

All experiments are run on a Linux server with Intel Xeon 2.83 GHz CPU and 32 GB memory. ISPD 05/06 placement benchmarks [21], [22] are used as the test circuits. For the ISPD 06 circuits, we use the density-penalty scaled HPWL defined in [22]. First, we show the comparison of various clustering algorithms for different clustering objectives. Second, we compare SCPlace with the stat-of-the-art placement algorithms.

A. Comparison of Clustering Algorithms

We compare SC with three clustering algorithms FC [13], BC [16], and NC [17]. We implemented FC and BC by ourselves and obtained the binary of NC from the authors [17]. For BC the lazy-update [16] is used to speed up its runtime.

In the experiments, the clustering algorithm is applied as a pre-processing step before placement (see Fig. 6). We adopt mPL6 [5] as the placer. Due to the same reason mentioned in Section V, we use flat-mPL6 here. In Fig. 6 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, flat-mPL6 detailed placer alone may not be enough to generate a good result. So we apply the detailed placer FastDP [6] to further refine the layout after flat-mPL6 detailed placement.

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 five sets of experiments.

1) Clustering targeting at safe cluster: we compare SC-G with FC and BC. FC and BC's target γ is set the same as SC-G's. Table IV 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.
2) 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 V, SC-R consistently generates the *best* HPWL for all 16 test cases, except 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× 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.
3) 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 VI 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, and 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. 7 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.
4) Clustering targeting at best placement wirelength: Table VI shows that various γ leads to various HPWL for each circuit. Here, we show that for most of the circuits, SC is able to automatically stop clustering, when the γ for the best HPWL is reached (see Table VII). Readers may compare Tables VI and VII to verify this. To see how one-level clustering compares with multilevel clustering, we generate the results of original multilevel mPL6 with FastDP ("mPL6+FastDP") in Table VII. The clustering time and final γ inside mPL6 are listed in Table VII. We can see that mPL6 has four levels of clustering and placement. Comparing SC with "mPL6+FastDP," even though SC on average generates 3% worse HWPL, for almost half of the circuits SC's HPWL is even better than "mPL6+FastDP." For most circuits, the HPWL generated by SC and "mPL6+FastDP" are very comparable. Regarding the total time, SC is significantly faster than "mPL6+FastDP" 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 that we see prospective improvements if SC is applied into the multilevel placement framework.

B. Comparison of Placement Algorithms

In this subsection, we compare SCPlace with the state-of-the-art placement algorithms.

TABLE IV

COMPARISON WITH FIRSTCHOICE AND BESTCHOICE BASED ON SC-G'S CLUSTERING RATIO

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	10 543	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	17 305	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	10 200	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	10 124	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	19 219	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**

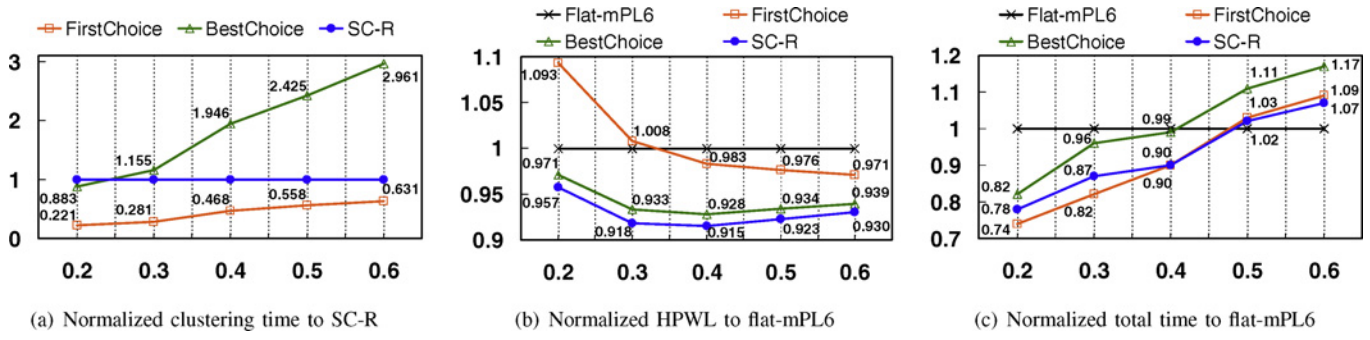
*: Comparison of scaled HPWL.

TABLE V

COMPARISON WITH FIRSTCHOICE, BESTCHOICE AND NETCLUSTER BASED ON NETCLUSTER'S CLUSTERING RATIO

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	10 543	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	17 305	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	10 200	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	10 124	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	19 219	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**

*: Comparison of scaled HPWL.



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

Fig. 7. Average normalized clustering time, HPWL, and total time over all circuits for target $\gamma = 0.2, 0.3, 0.4, 0.5, 0.6$. (a) Normalized clustering time to SC-R. (b) Normalized HPWL to flat-mPL6. (c) Normalized total time to flat-mPL6.

TABLE VI

COMPARISON WITH FIRSTCHOICE AND BESTCHOICE ON TARGET γ = 0.2, 0.3, 0.4, 0.5, 0.6

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	10 543	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	17 305	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	10 200	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	10 124	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	19 219	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

*: Comparison of scaled HPWL.

TABLE VII
COMPARISON WITH MULTILEVEL MPL6

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	10 516	8842	420	0.30	131	0.005	4
bigblue4	812.89	**803.43**	806.83	10 543	15 460	19 457	1622	0.33	468	0.008	4
adaptec5*	731.47	461.99	**429.97**	7892	5919	10 796	697	0.32	149	0.005	4
newblue1*	109.85	88.10	**64.72**	17 305	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**	10 200	14 986	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**	10 124	8405	16 220	625	0.34	251	0.008	4
newblue6*	520.95	**498.44**	516.43	7575	11 081	13 566	2059	0.33	255	0.009	4
newblue7*	1076.36	**1042.97**	1070.08	19 219	21 049	32 561	1159	0.34	278	0.014	4
Normalized	**1**	**0.910**	**0.879**	**1**	**1.086**	**1.412**					

*: Comparison of scaled HPWL.

TABLE VIII
COMPARISON WITH ORIGINAL MULTILEVEL MPL6

Circuit	HPWL ($\times 10e6$)		Total Time (s)		Clustering Info. Inside SCPlace				
					SafeChoice		Physical SafeChoice		
	mPL6	SCPlace	mPL6	SCPlace	Time (s)	γ	Time (s)	γ	# of Levels
adaptec1	78.05	**76.50**	1769	937	54	0.43	109	0.34	1
adaptec2	91.76	**86.30**	1940	1504	52	0.44	101	0.36	1
adaptec3	214.29	**204.10**	5949	2981	167	0.42	192	0.34	1
adaptec4	194.25	**183.20**	5487	2652	150	0.40	213	0.32	1
bigblue1	96.75	**93.58**	2158	1182	60	0.43	136	0.33	1
bigblue2	152.33	**144.39**	5842	3345	333	0.43	313	0.36	1
bigblue3	343.89	**336.01**	8382	7682	288	0.39	302	0.31	1
bigblue4	829.42	**790.76**	18 590	12 486	1219	0.42	1233	0.33	1
adaptec5*	430.42	**419.72**	10 714	5528	459	0.41	263	0.32	1
newblue1*	**73.21**	77.27	2489	10 798	218	0.41	55	0.36	1
newblue2*	201.63	**194.66**	7109	4642	54	0.70	79	0.61	1
newblue3*	284.04	**281.59**	9508	13 736	577	0.39	337	0.33	1
newblue4*	302.04	**295.98**	9410	4272	288	0.43	64	0.38	1
newblue5*	536.29	**522.71**	16 085	10 149	407	0.43	201	0.37	1
newblue6*	521.28	**494.10**	13 457	10 877	1481	0.42	1113	0.34	1
newblue7*	1083.66	**1035.15**	32 372	23 356	1003	0.43	932	0.34	1
Normalized	**1.036**	**1**	**1.549**	**1**					

*: comparison of scaled HPWL.

1) First, we compare SCPlace with mPL6. We run both algorithms on the same machine. The results and the clustering information inside SCPlace are shown in Table VIII. In terms of the HPWL, SCPlace is consistently better than mPL6, except for one circuit (i.e., newblue1). On average, SCPlace generates 4% better HPWL than mPL6. Regarding the total runtime, SCPlace is 55% faster than mPL6. As mentioned in Section V, in the first phase of SCPlace we set $C_t = 16$ rather than $C_t = 21$ for non-physical SafeChoice, so that the non-physical SafeChoice will stop clustering earlier to generate less low-quality clusters.
2) Second, we compare SCPlace with all other placement algorithms. Because some of the placers' binaries are not publicly available, instead of running every placer on the same machine, we directly cite the results from [7]. As far as we know, RQL [7] is the latest published placement algorithm in academic area, and it generates the best results on average compared with all previous placers. The experimental results are shown in Table IX. The "Previously Best" column shows the previously best HPWL achieved by other placers for each circuit. The results are quite promising. Regarding the HPWL, SCPlace is 2%, 25%, 9%, 6%, 3%, 21%, 6%, and 1% better than NTUplace3, Capo10.5, mFAR, APlace, mPL6, Dragon, Kraftwerk, and RQL, respectively. Even though SCPlace is 1% worse than the previously best approach, for 11 out of 16 circuits, SCPlace generates better results, which means SCPlace has broken the records for 11 circuits. All of the placers here, except for Kraftwerk, have at least four levels of placement. Using only one level of clustering and two phases of placement, SCPlace is able to beat all of them.

TABLE IX
HPWL COMPARISON WITH THE STATE-OF-THE-ART PLACEMENT ALGORITHMS

Circuit	NTUplace3 [1]	Capo10.5 [2]	mFAR [3]	APlace [4]	mPL6 [5]	Dragon [23]	Kraftwerk2 [24]	RQL [7]	Previously Best	SCPlace
adaptec1	80.93	91.28	—	78.35	77.91	—	82.43	77.82	77.82	**76.50**
adaptec2	89.95	100.75	91.53†	87.31†	91.96	94.72†	92.85	88.51	87.31	**86.30**
adaptec3	214.20	228.47	—	218.52	214.05	—	227.22	210.96	210.96	**204.10**
adaptec4	193.74	208.35	190.84†	187.65†	194.23	200.88†	199.43	188.86	187.65	**183.20**
bigblue1	97.28	108.60	97.70†	94.64†	96.79	102.39†	97.67	94.98	94.64	**93.58**
bigblue2	152.20	162.92	168.70†	143.82†	152.33	159.71†	154.74	150.03	**143.82**	144.39
bigblue3	348.48	398.49	379.95†	357.89†	344.37	380.45†	343.32	323.09	**323.09**	336.01
bigblue4	829.16	965.30	876.28†	833.21†	829.35	903.96†	852.40	797.66	797.66	**790.76**
adaptec5*	448.58	494.64	476.28	520.97	431.14	500.74	449.48	443.28	431.14	**419.72**
newblue1*	61.08	98.48	77.54	73.31	67.02	80.77	66.19	64.43	**61.08**	77.27
newblue2*	203.39	309.53	212.90	198.24	200.93	260.83	206.53	199.60	198.24	**194.66**
newblue3*	278.89	361.25	303.91	273.64	287.05	524.58	279.57	269.33	**269.33**	281.59
newblue4*	301.19	362.40	324.40	384.12	299.66	341.16	309.44	308.75	299.66	**295.98**
newblue5*	509.54	659.57	601.27	613.86	540.67	614.23	563.15	537.49	**509.54**	522.71
newblue6*	521.65	668.66	535.96	522.73	518.70	572.53	537.59	515.69	515.69	**494.10**
newblue7*	1099.66	1518.75	1153.76	1098.88	1082.92	1410.54	1162.12	1057.79	1057.79	**1035.15**
Normalized	**1.02**	**1.25**	**1.09**	**1.06**	**1.03**	**1.21**	**1.06**	**1.01**	**0.99**	**1**

* Comparison of scaled HPWL, † results are tuned for each circuit, the results of all other placers (except Kraftwerk2) are cited from [7], and "—" denotes unavailable results in [7].

TABLE X
RUNTIME BREAKDOWN OF SCPLACE

Steps in SCPlace	Runtime%
Non-physical clustering	6%
Global placement	39%
Physical clustering	6%
Incremental placement	45%
Detailed placement	4%

The runtime breakdown of SCPlace is presented in Table X. It shows that the total runtime is dominated by two steps, i.e., global placement and incremental placement. Both non-physical and physical clustering contribute only 6% of the total runtime.

VII. CONCLUSION

In this paper, we 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. Based on SafeChoice, we derived physical SafeChoice, and integrated it into a high-quality analytical placer, SCPlace. Promisingly, by a simple two-phase of placement, SCPlace significantly outperforms all state-of-the-art placement algorithms.

Our future work includes three directions: 1) to derive the safe condition for more than two vertices; 2) to develop our own placer based on SafeChoice, rather than feeding the clustered netlist to flat-mPL6 binary; and 3) to integrate SafeChoice into other algorithms, e.g., hypergraph partitioning. Regarding the last point, we can simply integrate SafeChoice into existing partitioner. Or more interestingly, we can propose a safe condition for hypergraph partitioning, e.g., what is the safe condition to do a partition?

Finally, the source code of SafeChoice is publicly available at [25].

ACKNOWLEDGMENT

The authors would like to thank G. Luo from the Computer-Aided Design Group, University of California, Los Angeles, and L. Rakai from the University of Calgary, Calgary, AB, Canada, for the help with mPL6 and NetCluster, respectively. They are also grateful to the anonymous reviewers for their helpful suggestions and comments on this paper.

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*, 2006, pp. 187–192.

[2] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov, "Min-cut floorplacement," *IEEE Trans. Comput.-Aided Design*, vol. 25, no. 7, pp. 1313–1326, Jul. 2006.

[3] B. Hu and M. Marek-Sadowska, "Multilevel fixed-point-addition-based VLSI placement," *IEEE Trans. Comput.-Aided Design*, vol. 24, no. 8, pp. 1188–1203, Aug. 2005.

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

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

[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*, 2007, pp. 135–140.

[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*, 2007, pp. 453–458.

[8] J. Z. Yan, N. Viswanathan, and C. Chu, "Handling complexities in modern large-scale mixed-size placement," in *Proc. DAC*, 2009, pp. 436–441.

[9] C. J. Alpert and A. B. Kahng, "Recent developments in netlist partitioning: A survey," *Integr. VLSI J.*, vol. 19, nos. 1–2, pp. 1–81, Aug. 1995.

[10] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," in *Proc. DAC*, 1997, pp. 526–529.

[11] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," in *Proc. DAC*, 1999, pp. 343–348.

[12] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel k-way hypergraph partitioning," in *Proc. DAC*, 1997, pp. 530–533.

[13] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. ISPD*, 2005, pp. 185–192.

[14] J. Cong and S. K. Lim, "Edge separability-based circuit clustering with application to multilevel circuit partitioning," *IEEE Trans. Comput.-Aided Design*, vol. 23, no. 3, pp. 346–357, Mar. 2004.

[15] B. Hu and M. Marek-Sadowska, "Fine granularity clustering-based placement," *IEEE Trans. Comput.-Aided Design*, vol. 23, no. 4, pp. 527–536, Apr. 2004.

[16] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng, "A fast hierarchical quadratic placement algorithm," *IEEE Trans. Comput.-Aided Design*, vol. 25, no. 4, pp. 678–691, Apr. 2006.
[17] J. Li, L. Behjat, and J. Huang, "An effective clustering algorithm for mixed-size placement," in *Proc. ISPD*, 2007, pp. 111–118.
[18] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. DAC*, 1982, pp. 175–181.
[19] J. Z. Yan, C. Chu, and W.-K. Mak, "SafeChoice: A novel clustering algorithm for wirelength-driven placement," in *Proc. ISPD*, 2010, pp. 185–192.
[20] H. Chen, C.-K. Cheng, N.-C. Chou, A. B. Kahng, J. F. MacDonald, P. Suaris, B. Yao, and Z. Zhu, "An algebraic multigrid solver for analytical placement with layout based clustering," in *Proc. DAC*, 2003, pp. 794–799.
[21] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 placement contest and benchmarks suite," in *Proc. ISPD*, 2005, pp. 216–220.
[22] G.-J. Nam, "ISPD 2006 placement contest: Benchmark suite and results," in *Proc. ISPD*, 2006, p. 167.
[23] T. Taghavi, X. Yang, B.-K. Choi, M. Wang, and M. Sarrafzadeh, "Dragon2005: Large-scale mixed-size placement tool," in *Proc. ISPD*, 2005, pp. 245–247.
[24] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2: A fast force-directed quadratic placement approach using an accurate net model," *IEEE Trans. Comput.-Aided Design*, vol. 27, no. 8, pp. 1398–1411, Aug. 2008.
[25] *SafeChoice Source Code* [Online]. Available: http://www.public.iastate.edu/∼zijunyan

Jackey Z. Yan received the B.S. degree in automation from the Huazhong University of Science and Technology, Wuhan, China, in 2006. He is currently pursuing the Ph.D. degree in computer engineering from the Department of Electrical and Computer Engineering, Iowa State University, Ames.

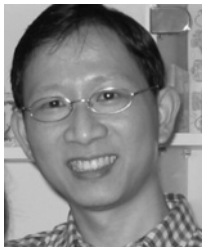
Since November 2010, he has been a Senior Technical Staff Member with the Placement Technology Group, Cadence Design Systems, Inc., San Jose, CA. His current research interests include very large scale integration physical designs, specifically in algorithms for floorplanning and placement, and physical synthesis integrated system-on-a-chip designs.

Mr. Yan's work on fixed-outline floorplanning was nominated for the Best Paper Award at the Design Automation Conference in 2008. Another of his works on hypergraph clustering for wirelength-driven placement was nominated for the Best Paper Award at the International Symposium on Physical Design in 2010. He received the Ultraexcellent Student Award from Renesas Technology Corporation, Tokyo, Japan, in 2005.

Chris Chu received the B.S. degree in computer science from the University of Hong Kong, Pokfulam, Hong Kong, in 1993, and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1994 and 1999, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames. His areas of expertise include computer-aided design of very large scale integration physical design, and design and analysis of algorithms. His current research interests include performance-driven interconnect optimization and fast circuit floorplanning, placement, and routing algorithms.

Dr. Chu received the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN (TCAD) Best Paper Award for his work on performance-driven interconnect optimization in 1999, another IEEE TCAD Best Paper Award for his work on routing tree construction in 2010, the ISPD Best Paper Award for his work on efficient placement algorithms in 2004, and the Bert Kay Best Dissertation Award from the Department of Computer Sciences, University of Texas, in 1998 and 1999.



Wai-Kei Mak received the B.S. degree in computer science from the University of Hong Kong, Pokfulam, Hong Kong, in 1993, and the M.S. and Ph.D. degrees in computer science from the University of Texas, Austin, in 1995 and 1998, respectively.

From 1999 to 2003, he was with the Department of Computer Science and Engineering, University of South Florida, Tampa, as an Assistant Professor. Since 2003, he has been with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, where he is currently an Associate Professor. His current research interests include very large scale integration physical design automation, field-programmable gate array architecture, and computer-aided design.

Dr. Mak has served on the program and/or the organizing committees of the Asia South Pacific Design Automation Conference, the International Conference on Field Programmable Logic and Applications, and the International Conference on Field-Programmable Technology. He was the General Chair of the 2008 International Conference on Field-Programmable Technology and was the Technical Program Co-Chair of the same conference in 2006. He has been in the Steering Committee of the International Conference on Field-Programmable Technology since 2009.