A Robust Min-Cut Improvement Algorithm based on Dynamic Look-ahead Weighting

Katsunori Tani
C&C Research Laboratories, NEC Corporation
Kawasaki, Kanagawa 216, Japan
e-mail: tani@sbl.cl.nec.co.jp

Abstract— This paper proposes an approach to enhance Fiduccia-Mattheyses' min-cut algorithm. The approach includes two new ideas: Look-ahead Weighting **and** Dynamic Weighting. **The former is based on the concept of VLSI placement method using quadratic programming. The latter is a technique to carry the better behavior of move-and-lock improvement strategy. Experiments on practical circuits with 5K∼140K cells show that the proposed approach achieves promising results.**

I. Introduction

Fabrication technology increases the circuit size to be designed very rapidly. Nowadays, a circuit with over 1M-gate, or over 100K-cell (function block), can be realized in one chip. MCM and rapid prototyping by FPGA's give a stimulus to this trend toward large-scale system. To fight against it, circuit partitioning becomes more important in most of design automation arena.

Circuit partitioning can be basically formulated as a graph-theoretical problem: Given a graph in which a node and an edge represent a cell and a connection (net) between cells, respectively, divide the graph into two parts so that the number of nets spreading both parts is minimum. The objective above is called **cut value**. If the problem has size limitation to each part, it is in the class of NP-hard (**bounded min-cut**) [1].

Kernighan and Lin proposed the first practical heuristic (KL-algorithm) [2] to the bounded min-cut problem. The heuristic improves the initial bipartitioning by repeating the swap of two cells (pairwise-interchanging) to reduce the cut value.

Fiduccia and Mattheyses improved KL-algorithm and proposed highly efficient heuristic [3]. It gave a nice framework for min-cut improvement, and many modifications have been brought up under the framework [4][5].

Another kind of improvement was done by Goldberg and Burstein [6]. They found that the results by iterative improvement strategy, such as KL-algorithm, depend heavily on the edge density, i.e., the ratio of the number of edges to the number of cells. They reported that KL-algorithm yields good result if the ratio is higher than five. However, VLSI circuit is usually very sparse. Accordingly, they suggested the edge-contractions to increase density and iterative improvement on the resultant dense graph.

Bipartitioning using analytical placement technique was recently proposed [7]. In the method, cells are roughly placed on a chip by solving quadratic programming problem, and the best cut position is sought by scanning the chip left-to-right or bottom-to-top[1]. This approach seems to produce better bisection than eigen-vector-based approach [8]. However, in comparison with graph-theoretic algorithm, the required CPU-time is fairly longer owing to intensive numerical computation.

This paper proposes a new algorithm under Fiduccia-Mattheyses's framework. It contains two new features: **look-ahead weighting** and **dynamic weighting**. The former is based on the concept of analytical VLSI placement using quadratic programming. The latter is a technique to carry the better behavior of conventional move-and-lock improvement strategy. Both of those ideas act in harmony toward the global optimum. Experimental results on practical circuits with 5K∼140K cells and the comparison between the proposed algorithm and previous works are also included.

II. Background

A. Definitions and Formulation

In this paper, we deal with the circuit of single VLSI chip. The following discussion is applicable to larger circuit at system-level design by regarding each module/macro as a cell.

VLSI circuit is represented by a hypergraph $\mathsf{G} = (\mathsf{C}, \mathsf{N})$. C and N correspond to the set of cells and the set of nets in the circuit, respectively. Note that a net $\mathsf{n} \in \mathsf{N}$ may be composed of three or more cells. The set of the nets connected to a cell $\mathsf{c} \in \mathsf{C}$ is denoted by $\mathsf{N}(\mathsf{c})$ $(\subseteq \mathsf{N})$. Similarly, the set of the cells connected to a net $\mathsf{n} \in \mathsf{N}$ is denoted by $\mathsf{C}(\mathsf{n})$ $(\subseteq \mathsf{C})$.

For each cell $\mathsf{c} \in \mathsf{C}$, we represent the size of c by $\mathsf{S}(\mathsf{c})$. As for a net, we should consider the significance of the net. $\mathsf{W}(\mathsf{n})$ denotes that of a net n (the net with larger $\mathsf{W}(\star)$-value is more significant). $\mathsf{S}(\star)$ and $\mathsf{W}(\star)$ are defined as positive integers. For the set of cells $\mathsf{C}' \subseteq \mathsf{C}$, $\mathsf{S}(\mathsf{C}')$ means the sum of $\mathsf{S}(\mathsf{c})$ for all $\mathsf{c} \in \mathsf{C}'$. $\mathsf{W}(\mathsf{N}')$ $(\mathsf{N}' \subseteq \mathsf{N})$ is defined in the same sense.

[1] Ref.[7] uses ratio cut measure rather than min-cut objective. However, the place-and-scan technique can be applied to conventional min-cut bipartitioning.

Given a bipartitioning $(C_1; C_2)$ ($C_1 \cup C_2 = C$, $C_1 \cap C_2 = \phi$) of G, the set $N(C_1; C_2)$ defined below and $W(N(C_1; C_2))$ are **cut** and **cut value** of the bipartitioning, respectively.

$$N(C_1; C_2) = \{n \mid n \in N,\ C(n) \cap C_1 \neq \phi,\ C(n) \cap C_2 \neq \phi\} \quad (1)$$

In addition, we say that the bipartitioning $(C_1; C_2)$ locates a cell $c \in C_1$ (C_2) at the left (right) side.

Using above definitions, bounded min-cut bipartitioning problem is formulated as follows:

Bounded Min-Cut Bipartitioning Problem

Given a hypergraph $G = (C, N)$ and real numbers $\alpha \in (0, 1)$ and $\beta \geq 0$, find the bipartitioning $(C_1; C_2)$ of G with minimum value $W(N(C_1; C_2))$, subject to the size balance constraint $| \alpha \cdot S(C) - S(C_1) | \leq \beta$.

α is the ratio of the target size of left half to the total size (it is usually set by 0.5). β is acceptable difference between the target size and the actual size of left half. In a VLSI placement application, there may exist some cells whose locations are fixed (e.g., external I/O buffers).

The remaining part of this section explains Fiduccia-Mattheyses' algorithm (FM-algorithm) and the related works under their framework.

B. Fiduccia-Mattheyses' Heuristic

Fiduccia and Mattheyses [3] modified KL-algorithm by moving a cell from current side to opposite side at a time. Their criterion to select the cell to be moved next is called **gain**. Gain is the attribute of each cell and defined as the decrease of cut value by moving the cell. For example, the gains of cells A and B of Fig.1 are +1 and +2, respectively. The gain of a cell c is denoted by $G_{FM}(c)$.

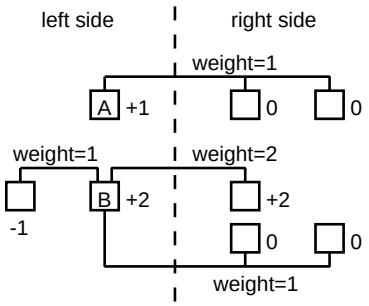

Fig. 1. Cell gain in FM-algorithm

Let $T(n, L)$ ($T(n, R)$) be the number of such terminals of a net n that they are located in the left (right) side. Then, the gain $G_{FM}(c)$ is expressed as follows:

$$G_{FM}(c) = \sum_{\substack{n \in N(c) \\ T(n,\, s(c)) = 1}} W(n) \ - \sum_{\substack{n \in N(c) \\ T(n,\, \overline{s(c)}) = 0}} W(n) \quad (2)$$

where $s(c)$ and $\overline{s(c)}$ stand for the current side (L or R) and the opposite side (R or L) of cell c, respectively. There holds the inequality below as to the range of $G_{FM}(c)$:

$$| G_{FM}(c) | \leq \sum_{n \in N(c)} W(n) \quad (3)$$

The outline of FM-algorithm is as follows (it assumes that an initial bipartitioning is given):

Step.1: Make all cells unlocked except for the cells with predefined location.

Step.2: Select the cell **from each side** which is not locked, has the highest gain in the side, and does not violate the balance constraint by moving it.

Step.3: If no cells can be selected, then exit.

Step.4: Choose the cell with higher gain among the selected at most two cells, breaking ties by choosing the one which gives better balance.

Step.5: Swap the side of the cell which is finally selected at Step.4, and lock it.

Step.6: Update gains of the cells which are not locked.

Step.7: Repeat Steps 2 through 6 until exit at Step.3.

After these seven steps (called **one-pass**) are performed, we can try one or more passes to the best bipartitioning obtained thus far to improve it still more. The iteration is called **multi-pass** FM-algorithm.

Steps 2 and 6 are the dominant processes as to computational time. Fiduccia and Mattheyses managed the gains of cells on each side using bucket structure. Note that gain is an integer which satisfies the inequality (3). This data structure makes the cell selection and the update of buckets be performed in constant time per move. Re-calculating gains can be efficiently done since only the cells which are adjacent to the moved cell require their gains to be updated. Ref.[3] proved that one-pass takes the time proportional to the total number T of terminals in given circuit ($T = \sum_{n \in N} | C(n) |$).

C. High-Order Gain

Krishnamurthy [4] pointed out the randomness of cell selection in FM-algorithm: The definition of $G_{FM}(\star)$ yields many cells with same gain, and hence, the selection of best cell is done somewhat arbitrarily. Fig.2 illustrates his observation. The gains of both cells A and B of Fig.2 are +1. However, the cell B is preferable choice since B's move will lead the cell C to be moved in subsequent process (A's move does not directly lead other cells).

He resolved the randomness by introducing **gain vector** $G_{KR}(c) = \langle G^{(1)}_{KR}(c), G^{(2)}_{KR}(c), \cdots, G^{(k)}_{KR}(c) \rangle$ for each cell c. The first-order gain $G^{(1)}_{KR}(c)$ equals to $G_{FM}(c)$. The subsequent i'th-order gain $G^{(i)}_{KR}(c)$ ($2 \leq i \leq k$) indicates how much the cell c will assist other cells in moving to decrease cut value. Namely, it is a kind of look-ahead gain. Consider the cell B of Fig.2. After moving it to right side, $G^{(1)}_{KR}(C)$ will be changed from 0 to +1. In other words, B'move indirectly decreases cut value by 1 via the cell C, and hence, $G^{(2)}_{KR}(B)$ is +1. $G^{(i)}_{KR}(\star)$ is formally defined as follows:

$$G^{(i)}_{KR}(c) = \sum_{\substack{n \in N(c) \\ \beta(n,\, s(c)) = i \\ \beta(n,\, \overline{s(c)}) > 0}} W(n) \ - \sum_{\substack{n \in N(c) \\ \beta(n,\, s(c)) > 0 \\ \beta(n,\, \overline{s(c)}) = i - 1}} W(n) \quad (4)$$

where $\beta(n, s)$, called **binding number**, represents how tightly the net n is bound to the side s and is defined using the number $\alpha_{locked}(\alpha_{unlocked})$ = the number of the terminals of n each of which belongs to a locked (unlocked) cell on side s:

$$\beta(n, s) = \begin{cases} \alpha_{unlocked} & \text{if } \alpha_{locked} = 0 \\ \infty & \text{otherwise} \end{cases} \quad (5)$$

He suggested distinguishing the effects of cells using $G_{KR}(\star)$ (the order of two vectors is defined in lexicographical manner) and concluded that the time complexity of FM-algorithm with $G_{KR}(\star)$ is $O(k \cdot T)$ and a small number between 2 and 3 is sufficient for the value of k.

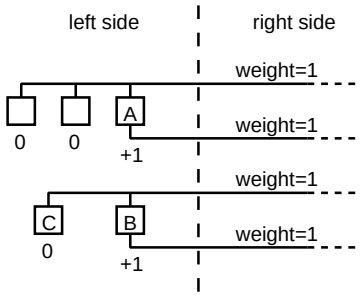

Fig. 2. Randomness induced by FM-gains

D. Net-Crossing Model

Sechen and Chen [5] proposed an objective function to calculate gains. As stated at Section IIA, cut value is determined by whether a net is cut or not rather than how the net is cut. Let us consider a 4-terminal net n. If n is cut, the distribution of n's terminal is one of the followings: (1;3), (2;2), (3;1), where p (q) of the notation $(p;q)$ represents the number of terminals on the left (right) side.

Their objective function is **expected number of crossings**. Let us explain the concept using Fig.3 which illustrates a terminal distribution of a 10-terminal net. Since there are 3 terminals on the left side, the right side is divided into 3+1=4 regions (each region is called **bin**). Each of 10-3=7 terminals on the right side belongs to one of bins. The number of crossings is defined as the number of inter-side connections. For example, the terminal distribution of Fig.3 has four crossings[2].

They have calculated all possible patterns of terminal distributions and probability of each pattern to derive the following formula [3] for computing the expected number of crossings, $X_{exp}(p;q)$, in the case of $(p;q)(p \le q)$ distribution (see [5] as to detailed derivation process):

$$X_{exp}(p;q) = \frac{X_o + X_e}{\sum_{b=1}^{p+1} \frac{(p+1)!}{b!\,(p+1-b)!} \cdot \frac{(q-1)!}{(b-1)!\,(q-b)!}} \quad (6)$$

[2]It is assumed that if a right cell c is in a bin #i, there are crossings between c and the left cells nipping the bin #i.

[3]Eqns.(15) and (16) in [5]

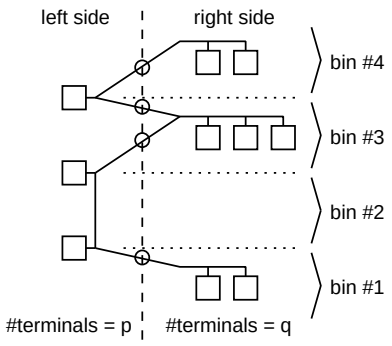

Fig. 3. Expected number of crossings

$$X_o = \sum_{\substack{c=1 \\ c \in \text{odd numbers}}}^{2p-1} 2c \cdot \frac{(q-1)!}{\lfloor \frac{c}{2} \rfloor!\,(q-1-\lfloor \frac{c}{2} \rfloor)!} \cdot \frac{(p-1)!}{\lfloor \frac{c}{2} \rfloor!\,(p-1-\lfloor \frac{c}{2} \rfloor)!} \quad (7)$$

$$X_e = \sum_{\substack{c=2 \\ c \in \text{even numbers}}}^{2p} c\,(p+q-c) \cdot \frac{(q-1)!}{\frac{c}{2}!\,(q-\frac{c}{2})!} \cdot \frac{(p-1)!}{(\frac{c}{2}-1)!\,(p-\frac{c}{2})!} \quad (8)$$

Eqn.(6) was derived on the assumption that the number of bins is much larger than p. They refined Eqn.(6) in the case that the assumption is not valid (see Eqns.(18) and (19) in [5]). Using $X_{exp}(p;q)$, Sechen-Chen's gain $G_{SC}(c)$ of a cell c is expressed as follows:

$$G_{SC}(c) = \sum_{n \in N(c)} W(n) \cdot \{\tilde{X}_{exp}(T(n, s(c)); T(n, \overline{s(c)})) - \tilde{X}_{exp}(T(n, s(c)) - 1; T(n, \overline{s(c)}) + 1)\} \quad (9)$$

where

$$\tilde{X}_{exp}(p;q) = \begin{cases} 0 & \text{if } p \le 0 \text{ or } q \le 0 \\ X_{exp}(p;q) & \text{if } p \le q \text{ and } p > 0 \\ X_{exp}(q;p) & \text{otherwise} \end{cases} \quad (10)$$

Although $G_{SC}(c)$ is a real number, applying an appropriate scaling permits the FM-framework to be available. Ref.[5] concluded that defining the gain by the summation $G_{FM}(\star) + G_{SC}(\star)$ brought a good result. The computational complexity of FM-algorithm with $G_{SC}(\star)$ is not changed if we calculate $G_{SC}(\star)$ and store the values to a gain table in advance.

III. Dynamic Look-ahead Weighting

This section presents newly developed techniques. We begin with our strategy and net modeling, and then propose the techniques to obtain a robust improvement and promising results.

A. Strategy and Clique Model

The original KL-algorithm [2] was developed for an ordinary graph, that is, the degree of each net is assumed to be exactly two. On this assumption, each multi-terminal net must be converted into the set of 2-terminal nets by some pre-processing. To avoid the conversion, Schweikert

and Kernighan [9] suggested a hypergraph representation as a proper modeling of circuit, which was one of the basis of FM-algorithm. Since then, the representation has been the main current in the field of min-cut algorithm.

However, at the same time, it has been pointed out that the iterative improvement strategies such as pairwise-interchanging of KL-algorithm and single-cell-move of FM-algorithm do not bring a good result for a sparse graph. VLSI circuit is usually very sparse. The edge contraction technique [6] aims at the augmentation of cell-degree $|\,N(\star)\,|$. Although hypergraph represents a given circuit accurately, the cell-degree $|N(\star)|$ decreases to create a pitfall for iterative improvement as a result. Our observation is that 2-terminal net modeling is not accurate, however, it can give a global orientation. Consequently, we apply two-phase optimization, which is outlined as follows:

Phase.1 (global optimization)

Transform each multi-terminal net of a given circuit into the set of weighted 2-terminal nets, and then minimize cut value of the converted circuit.

Phase.2 (local optimization)

Using the resultant bipartitioning of Phase.1 as initial solution, minimize cut value on the proper circuit by conventional FM-algorithm.

We use a simple clique model at Phase.1, or complete graph model, to represent each net. For t-terminal net, the weight ratio $\frac{2}{t}$ is attached to each edge (pair of terminals) of the corresponding clique. For example, a 4-terminal net n is converted into the six 2-terminal nets each of which has weight $W(n) \cdot \frac{2}{4}$.

Let us consider the terminal distribution of t-terminal net n. On the assumption that the net n is cut, there are $(t-1)$ types of distribution: (1;t-1), (2;t-2), $\cdots$, (t-1;1). It is clear that i'th-type of distribution "(i;t − i)" includes $i \cdot (t-i)$ cut edges on clique model. The function $F(i) = i \cdot (t-i)$ is symmetric upper-convex and has the maximum at $i = \frac{t}{2}$. Since $F(i)$ indicates the cut value of n on the clique model, we can use $F(i)$ to calculate a gain $G(c)$ of cell c:

$$G(c) = \sum_{n \in N(c)} W(n) \cdot X(c,n) \qquad (11)$$

where $X(c,n)$ is the gain per unit net weight with respect to net n, and it is expressed as follows using $T(n)$ which denotes the number of terminals of n:

$$X(c,n) = \frac{2}{T(n)} \{F(T(n,s(c))) - F(T(n,s(c)) - 1)\} \qquad (12)$$

$$= \frac{2}{T(n)} \{T(n,\overline{s(c)}) - T(n,s(c)) + 1\} \qquad (13)$$

Surprisingly, the gain $G(c)$ has well-coincidence with the gain $G_{SC}(c)$ (#bins $= \infty$). The fact is proven by Table I which shows the gains for some terminal distributions of a net with unit weight (for the cases #bins = 100 and 50, we calculate the refined $G_{SC}(\star)$ which is defined by Eqns. (18) and (19) in [5]).

Therefore, it can be said that our weighted clique modeling has at least same potential with Sechen-Chen's modeling. As a matter of course, if we only evaluate the gain $G(\star)$, then the clique-conversion of given hypergraph is not needed (it is sufficient to replace the values in a gain table with $G(\star)$). The next section describes the true worth of using the clique modeling.

TABLE I
COMPARISON OF $G(c)$ AND $G_{SC}(c)$ WITH RESPECT TO A NET WITH UNIT WEIGHT ($\#f = T(n, s(c))$, $\#t = T(n, \overline{s(c)})$)

$T(n)$	$\#f : \#t$	$G(c)$	$G_{SC}(c)$ for specified #bins		
			∞	100	50
2	1 : 1	1.00000	1.00000	1.00000	1.00000
3	1 : 2	1.33333	1.33333	1.33333	1.33333
4	1 : 3	1.50000	1.50000	1.50000	1.50000
4	2 : 2	0.50000	0.34615	0.33600	0.32604
5	1 : 4	1.60000	1.60000	1.60000	1.60000
5	2 : 3	0.80000	0.80000	0.78218	0.76471
6	1 : 5	1.66667	1.66667	1.66667	1.66667
6	2 : 4	1.00000	1.00000	0.97888	0.95817
6	3 : 3	0.33333	0.28415	0.27267	0.26164
7	1 : 6	1.71429	1.71429	1.71429	1.71429
7	2 : 5	1.14286	1.14286	1.11928	1.09617
7	3 : 4	0.57143	0.57143	0.55001	0.52951
8	1 : 7	1.75000	1.75000	1.75000	1.75000
8	2 : 6	1.25000	1.25000	1.22454	1.19958
8	3 : 5	0.75000	0.75000	0.72277	0.69673
8	4 : 4	0.25000	0.23577	0.22284	0.21061
9	1 : 8	1.77778	1.77778	1.77778	1.77778
9	2 : 7	1.33333	1.33333	1.30638	1.27996
9	3 : 6	0.88889	0.88889	0.85685	0.82624
9	4 : 5	0.44444	0.44444	0.42185	0.40063

B. Look-ahead Weighting

In the field of VLSI placement, analytical approaches have been investigated recently [10] [11] [12]. The approaches decide the placement of cells by solving quadratic programming problems. The circuit partitioning method using such an analytical placement program has been also proposed [7].

Let (X_i, Y_i) and C_{ij} $(= C_{ji} \geq 0)$ be the coordinate of cell i and the connection strength between cells i and j, respectively. Then, we can write the basic quadratic programming problem as follow (note that the coordinates (X_k, Y_k)'s are fixed for some cells such as I/O buffers):

$$\text{minimize} \quad F = \frac{1}{2} \sum_{i,\, j\,(i \neq j)} C_{ij} \{(X_i - X_j)^2 + (Y_i - Y_j)^2\} \qquad (14)$$

There hold the following equations with respect to the solution $(X_i^\star, Y_i^\star)$ of the problem:

$$X_i^\star = \frac{\sum_{j\,(\neq i)} C_{ij} X_j^\star}{\sum_{j\,(\neq i)} C_{ij}} \quad \text{and} \quad Y_i^\star = \frac{\sum_{j\,(\neq i)} C_{ij} Y_j^\star}{\sum_{j\,(\neq i)} C_{ij}} \qquad (15)$$

The pair of equations (15) means that the coordinate of a cell coincides with the gravity center of the coordinates of related cells.

Since the problem above is independent as to the two axes, we mention only X-axis in the following.

Let us consider the relation between connection strength $C_{\star\star}$ and the existence of X_i. In other words, if the

cell i is deleted with the connections to i, how we change the values of $C_{jk\,(j,\,k\neq i)}$ to keep the solution unchanged? – Solving the equation below symbolically answers the question ($\tilde{C}_{\star\star}$ represents the new values of $C_{\star\star}$):

$$\sum_{j\,(\neq i)} C_{ij}\,(X^\star_j - X^\star_i)^2 \;=\; \frac{1}{2} \sum_{\substack{j,\ k\,(\neq j)\\ j,\ k\neq i}} (\tilde{C}_{jk} - C_{jk})\,(X^\star_j - X^\star_k)^2 \qquad (16)$$

The answer is:

$$\tilde{C}_{jk\,(j,\,k\neq i)} \;=\; C_{jk} + \frac{C_{ij}\,C_{ik}}{\sum_{m\,(\neq i)} C_{im}} \qquad (17)$$

Proof
Substituting $X^\star_i$ in LHS of (16) using (15) produces

$$\frac{1}{(\sum_{m\,(\neq i)} C_{im})^2} \sum_{j\,(\neq i)} C_{ij}\,(X^\star_j \sum_{k\,(\neq i)} C_{ik} - \sum_{k\,(\neq i)} C_{ik}\,X^\star_k)^2 \qquad (18)$$

For any $a(\neq i)$ and $b(\neq i, \neq a)$, the coefficients of ${X^\star_a}^2$ and $X^\star_a\,X^\star_b$ of (18) are

$$\frac{C_{ia}\sum_{m\,(\neq i,a)} C_{im}}{\sum_{m\,(\neq i)} C_{im}} \quad \text{and} \quad -\frac{2\,C_{ia}\,C_{ib}}{\sum_{m\,(\neq i)} C_{im}} \qquad (19)$$

respectively. Also, the coefficients of ${X^\star_a}^2$ and $X^\star_a\,X^\star_b$ in RHS of (16) are

$$\sum_{m\,(\neq i,a)} (\tilde{C}_{am} - C_{am}) \quad \text{and} \quad -2\,(\tilde{C}_{ab} - C_{ab}) \qquad (20)$$

respectively. Hence, Eqn.(17) holds by the equality of the latters of (19) and (20). On the other hand, substituting $\tilde{C}_{am}$ of the former of (20) using (17) produces

$$\sum_{m\,(\neq i,a)} (\tilde{C}_{am} - C_{am}) \;=\; \sum_{m\,(\neq i,a)} \frac{C_{ia}\,C_{im}}{\sum_{n\,(\neq i)} C_{in}} \qquad (21)$$

$$\;=\; \frac{C_{ia}\sum_{m\,(\neq i,a)} C_{im}}{\sum_{n\,(\neq i)} C_{in}} \qquad (22)$$

and the RHS coincides with the former of (19). □

Eqn.(17) implies that the effect of a cell A to a cell B can be estimated even if the cell A is not adjacent to the cell B, i.e., $C_{AB}=0$. Fig.4 illustrates the observation. In the figure, the cell i is affected directly from j (1st-order cell) and is also affected indirectly from k and k′ (2nd-order cells) via j. We can estimate such indirect effects using (17) by assuming that the 1st-order cell j is deleted. Needless to say, actual deletion is not needed: Eqn.(17) indicates the graph configuration after deletion. It can be said that Eqn.(17) implies one of degree augmentation techniques[4] and another kind of high-order gain. Krishnamurthy's high-order gain of Section IIC reduces the randomness of cell selection by vectorization, however, the ordering vectors brings another problem: Lexicographical ordering is not always the best as he pointed by himself (see [4]).

Fig.5 gives the procedure to calculate the gain based on Eqn.(17) for a cell c. Note that the location of 1st-order cell (p in Fig.5) does not participate in the calculation of the gain. We discuss the treatment of their locations in the next section.

[4] Note that Eqn.(17) is equivalent to star-delta transformation of resistive network theory.

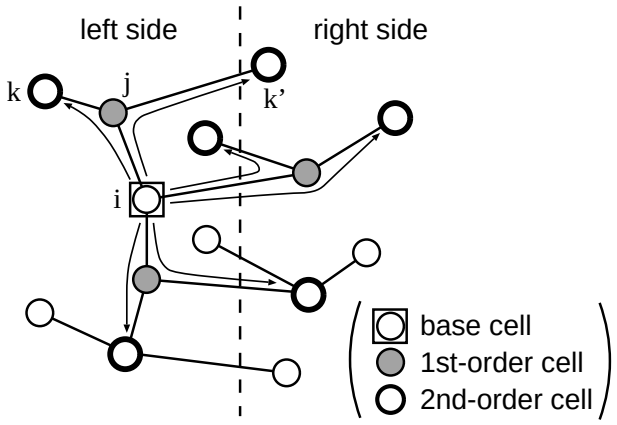

Fig. 4. Illustration of look-ahead Weighting

```
L01:  G_L(c) ← 0;
L02:  for each adjacent cell p of c {
L03:    S ← Σ_q C_pq;
        /* S is sum of connection weights related with p */
L04:    for each adjacent cell q(≠ c) of p {
L05:      if the current side of q differs from that of c
L06:        G_L(c) ← G_L(c) + C_cp C_pq / S;
L07:      else
L08:        G_L(c) ← G_L(c) − C_cp C_pq / S;
L09:    }
L10:  }
```

Fig. 5. Gain calculation based on look-ahead weighting

C. Dynamic Weighting for Locked Cells

As explained at Section IIB, FM-algorithm locks the cell which is moved once (locked cells never moves during the current pass). Let us pay attention to Fig.4 again. If the weight produced by the path $i-j-k'$ is large, the cell i will move to the right side. As a result, the cell j suffers strong force via the cells i and k′, and will go along with the cell i if j is not locked. This "chain-reaction" is the aim of look-ahead weighting.

However, if the cell j is locked, the observation above is turned to the other way. Since the path $i-j-k'$ is cut at j, the cell i should suffers no force from the cell k′. On the contrary, i should be affected by j directly. Fig.6 illustrates the concept. The weighting based on the concept heavily depends on a move-and-lock strategy. Hence, we say it "dynamic". On the other hand, look-ahead weighting is "static" since it depends on a constant topology.

The static gain calculation of Fig.5 can be easily modified into the dynamic version, which is shown in Fig.7.

D. Computational Complexity

Under the FM-framework, the following procedures are dominant as to time complexity: (1) calculating initial gain, (2) selecting the best cell and (3) updating gain after the move of a cell. In addition, we should consider

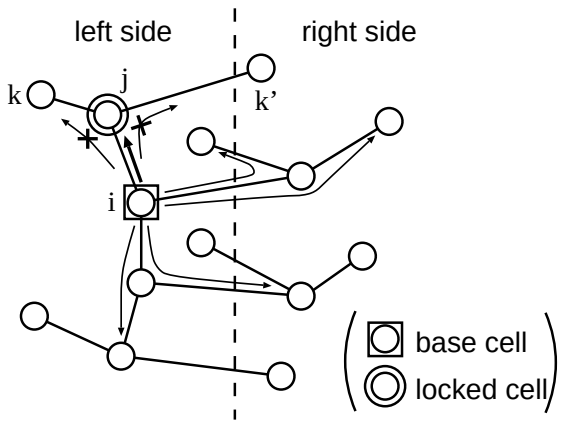

Fig. 6. Illustration of dynamic Weighting

```
DL01:  G_DL(c) ← 0;
DL02:  for each adjacent cell p of c {
DL03:    if p is locked {
DL04:      if the current side of p differs from that of c
DL05:        G_DL(c) ← G_DL(c) + C_cp;
DL06:      else
DL07:        G_DL(c) ← G_DL(c) − C_cp;
DL08:    } else {
DL09:      S ← Σ_q C_pq;
           /* S is sum of connection weights related with p */
DL10:      for each adjacent cell q(≠ c) of p {
DL11:        if the current side of q differs from that of c
DL12:          G_DL(c) ← G_DL(c) + C_cp C_pq / S;
DL13:        else
DL14:          G_DL(c) ← G_DL(c) − C_cp C_pq / S;
DL15:      }
DL16:    }
DL17:  }
```

Fig. 7. Gain calculation based on dynamic look-ahead weighting

(4) constructing graph from input specification, since we use clique modeling of a circuit.

The procedure (4) is theoretically dominant since on a clique modeling of hypergraph $G = (C, N)$ the number of edges is $O(|C|^2)$. However, the sparsity of VLSI circuit reduces the complexity close to $O(|C|)$ in practice.

The same observation can be applied to the procedures (1) and (3) (the gain based on the proposed weighting can be updated by incremental manner, which is similar to that of the original FM-algorithm). For the procedure (2), bucket structure is available as well as FM-algorithm by applying an appropriate scaling to convert real-valued gain into integer. Therefore the time complexity of the procedure (2) is the same with that of FM-algorithm.

As a result, time complexity of single pass execution of our algorithm is theoretically of quadratic order, however, close to linear in practice. It is the same with respect to space complexity.

IV. Experimental Results

We have implemented and tested the proposed algorithm, original FM-algorithm and the enhanced FM-algorithm by Sechen-Chen (SC-algorithm) on an NEC EWS4800/360AD (149MIPS). Table II summarizes the characteristics of practical circuits used for the tests. The data "D12" contains 144K-cells, which may be the largest circuit reported in published papers. Each test was performed in multi-pass style (we repeated passes on each phase until no improvement was attained). The margin of size balance constraint, β defined in Section IIA, was set by the maximum cell size in each circuit.

We performed two types of tests. The one is to improve the bipartitioning obtained by the min-cut algorithm of [13]. The algorithm uses hierarchical clustering and pairwise interchanging. Therefore, the resulting bipartitioning may be good. This type of test indicates the effectiveness of each algorithm. Table III shows the results. The character "H" ("V") which follows a data name means that the bipartitioning is done using a horizontal (vertical) cut-line (only the fixed-locations of I/O buffers depend on cut-line direction). The columns named "improve" and "CPU" are the improvement ratio $\frac{\text{cut value obtained by [13] - result}}{\text{cut value obtained by [13]}}$ and the total CPU-time required to update gains and bucket structures, respectively. The star mark in each row represents which algorithm yields the best solution.

The other is to improve the random bipartitioning. We thoroughly shuffled the result of [13] for each circuit keeping the total cell size of each side. This type of test indicates the robustness of each algorithm. Table IV shows the results. The items of each column and marks are the same with Table III.

Both tables prove the advantage of the proposed algorithm. Most of the star marks are in the columns for our algorithm. Especially, it can be said that our algorithm is highly robust. Notwithstanding a flat partitioning, some of the results of our algorithm have much lower cut value than the hierarchical approach. Also, computational time is not so longer than other algorithms.

On these tests, our two-phase optimization strategy showed interesting phenomenon: Although the cut value on hypergraph model after Phase.1 was slightly larger than that obtained by FM-algorithm [5], FM-algorithm of Phase.2 could improve the intermediate result to obtain the final result. The reason is not made clear yet, however, we think that our clique modeling give a good orientation to global optimum.

We have also implemented Krishnamurthy's algorithm and tested using the circuits of Table II. For each test, we observed that (1) resultant cut value is between the ones obtained by FM-algorithm and SC-algorithm (closer to the result of FM-algorithm rather than SC-algorithm) and (2) CPU-time is almost same as FM-algorithm within the range of 5%.

[5] Note that we minimize the cut value of weighted graph after the conversion for clique-modeling in Phase.1.

TABLE II
CHARACTERISTICS OF CIRCUITS USED FOR EXPERIMENTS
(C_m = MAX $|C(\star)|$, N_m = MAX $|N(\star)|$)

name	#cells	#nets	#pins	min : max : av. $S(\star)$	C_m	N_m
D1	5,024	5,417	17,311	1 : 35 : 6.79	93	14
D2	5,472	5,551	17,893	1 : 35 : 6.23	93	14
D3	5,124	9,031	29,416	1 : 38 : 6.72	42	19
D4	7,624	10,087	33,443	1 : 38 : 4.52	65	19
D5	7,484	10,436	36,344	1 : 28 : 4.99	98	11
D6	14,286	14,574	48,457	1 : 28 : 2.62	99	9
D7	18,323	24,953	86,592	1 : 19 : 4.90	66	11
D8	34,669	35,660	114,618	1 : 19 : 2.59	99	9
D9	37,809	57,691	161,171	1 : 52 : 5.57	32	25
D10	84,610	93,063	247,293	1 : 52 : 2.49	32	25
D11	82,605	126,927	361,620	1 : 90 : 5.31	58	37
D12	144,033	154,038	461,872	1 : 90 : 3.05	58	37

TABLE III
EXPERIMENTAL RESULTS
(IMPROVEMENT OF GIVEN BIPARTITIONING)

data	FM-algorithm		SC-algorithm		new algorithm	
	improve [%]	CPU [sec]	improve [%]	CPU [sec]	improve [%]	CPU [sec]
D1(H)	⋆ 4.24	0.36	2.54	0.71	3.39	4.19
D1(V)	⋆ 13.27	0.84	10.18	0.85	11.06	2.14
D2(H)	⋆ 4.76	0.71	3.46	0.66	⋆ 4.76	5.16
D2(V)	8.33	0.32	6.48	1.12	⋆ 9.26	3.22
D3(H)	6.09	0.74	⋆ 6.34	1.59	5.84	5.53
D3(V)	4.80	0.64	6.73	1.54	⋆ 7.69	5.57
D4(H)	1.11	0.66	2.66	0.77	⋆ 7.30	9.42
D4(V)	10.19	1.39	12.14	1.61	⋆ 15.05	12.13
D5(H)	0.80	0.65	0.80	1.46	⋆ 3.39	26.99
D5(V)	7.06	1.22	5.74	2.16	⋆ 13.69	15.46
D6(H)	14.39	3.50	23.99	4.58	⋆ 24.17	25.45
D6(V)	10.99	3.37	17.40	5.22	⋆ 31.69	23.16
D7(H)	14.23	1.86	20.41	6.55	⋆ 34.25	50.50
D7(V)	1.95	3.10	7.72	8.42	⋆ 12.96	76.58
D8(H)	16.37	4.95	⋆ 36.35	7.39	14.79	88.80
D8(V)	17.06	7.49	24.82	11.88	⋆ 31.02	76.29
D9(H)	14.50	7.01	19.59	16.38	⋆ 22.04	25.90
D9(V)	11.95	9.43	14.21	18.12	⋆ 15.71	20.78
D10(H)	19.56	20.38	21.89	43.55	⋆ 28.89	53.86
D10(V)	4.20	30.84	5.69	29.79	⋆ 7.87	44.72
D11(H)	0.00	3.44	0.00	7.28	0.00	21.01
D11(V)	0.00	2.92	0.00	5.85	0.00	30.65
D12(H)	0.00	5.56	0.00	11.91	0.00	28.64
D12(V)	7.33	12.85	7.33	26.94	⋆ 24.63	95.48
average	8.05	—	10.69	—	⋆ **13.73**	—

TABLE IV
EXPERIMENTAL RESULTS
(IMPROVEMENT OF RANDOM BIPARTITIONING)

data	FM-algorithm		SC-algorithm		new algorithm	
	improve [%]	CPU [sec]	improve [%]	CPU [sec]	improve [%]	CPU [sec]
D1(H)	-17.37	1.59	-41.53	2.00	⋆ 22.88	8.12
D1(V)	-27.43	0.93	-10.18	2.14	⋆ 3.10	7.90
D2(H)	6.06	1.74	20.80	1.80	⋆ 22.51	4.27
D2(V)	⋆ 20.37	2.32	-9.72	1.56	18.52	4.22
D3(H)	-127.41	0.96	-97.97	2.79	⋆ -27.61	9.27
D3(V)	-91.73	1.71	-37.12	3.04	⋆ -14.23	7.26
D4(H)	-119.25	1.84	-108.19	3.37	⋆ -47.12	21.26
D4(V)	-54.69	2.01	-60.19	2.83	⋆ -7.12	14.25
D5(H)	-46.11	1.96	-52.10	3.94	⋆ 8.98	24.20
D5(V)	-89.85	1.25	⋆ 0.22	4.91	-11.48	16.99
D6(H)	-22.88	5.40	⋆ -0.19	11.96	-3.32	35.70
D6(V)	⋆ 19.60	9.98	-65.57	7.08	16.67	29.82
D7(H)	-32.68	11.26	-23.16	9.81	⋆ 31.60	63.23
D7(V)	-65.13	13.14	-28.13	12.68	⋆ 20.67	63.61
D8(H)	-229.58	24.55	-26.06	15.14	⋆ 14.61	90.07
D8(V)	-38.99	31.46	-28.88	21.23	⋆ 9.75	65.32
D9(H)	-166.41	26.94	-76.79	27.06	⋆ -12.48	44.44
D9(V)	-140.83	28.86	-82.97	54.15	⋆ -9.23	53.28
D10(H)	-249.33	59.56	-236.44	95.54	⋆ -16.67	63.87
D10(V)	-105.34	59.32	-188.80	82.70	⋆ -11.08	52.60
D11(H)	-1105.14	22.81	-678.08	67.10	⋆ -7.95	99.52
D11(V)	-550.67	29.90	-375.20	56.35	⋆ 0.00	98.97
D12(H)	-1406.19	52.73	-1914.09	202.74	⋆ -27.84	124.91
D12(V)	-698.24	71.52	-1928.74	96.82	⋆ -60.11	209.45
average	-222.47	—	-252.05	—	⋆ **-3.62**	—

V. CONCLUSION

We have proposed the min-cut improvement algorithm based on dynamic look-ahead weighting. The look-ahead weighting was based on the concept of analytical placement technique using quadratic programming, and also included the concepts of degree augmentation and high-order gain. The dynamic weighting took the behavior of move-and-lock strategy into account and utilized it. The experimental results have proven the advantage, in particular, the robustness of the proposed algorithm.

REFERENCES

[1] M.R.Garey and D.S.Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H.Freeman and Company, 1979.

[2] B.W.Kernighan and S.Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," Bell Systems Technical Journal, vol.49, no.2, pp.291-307, 1970.

[3] C.M.Fiduccia and R.M.Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," Proc. DAC'82, pp.175-181.

[4] B.Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks," IEEE Trans. on Computers, vol.C-33, no.5, pp.438-446, 1984.

[5] C.Sechen and D.Chen, "An Improved Objective Function for Mincut Circuit Partitioning," Proc. ICCAD'88, pp.502-505.

[6] M.K.Goldberg and M.Burstein, "Heuristic Improvement Technique for Bisection of VLSI Networks," Proc. ICCD'83, pp.122-125.

[7] B.M.Riess, K.Doll and F.M.Johannes, "Partitioning Very Large Circuits using Analytical Placement Techniques," Proc. DAC'94, pp.646-651.

[8] L.Hagen and A.B.Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering," Proc. ICCAD'91, pp.10-13.

[9] D.G.Schweikert and B.W.Kernighan, "A Proper Model for the Partitioning of Electrical Circuits," Proc. Design Automation Workshop '79, pp.57-62.

[10] J.M.Kleinhans, G.Sigl, F.M.Johannes and K.J.Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," IEEE Trans. CAD, vol.10, no.3, pp.356-365, 1991.

[11] G.Sigl, K.Doll and F.M.Johannes, "Analytical Placement: A Linear or a Quadratic Objective Function?," Proc. DAC'91, pp.427-432.

[12] R.-S.Tsay, E.S.Kuh and C.-P.Hsu, "PROUD: A Sea-of-Gates Placement Algorithm," IEEE Design & Test of Computers, pp.44-56, December, 1988.

[13] M.Edahiro and T.Yoshimura, "New Placement and Global Routing Algorithms for Standard Cell Layouts," Proc. DAC'90, pp.642-645.