Partitioning Using Second-Order Information and Stochastic-Gain Functions[*]

Shantanu Dutt[1] and Halim Theny[2]
[1] EECS Dept., Univ. of Illinois at Chicago [2] Intel Corp., Folsom, CA

Abstract

A probability-based partitioning algorithm, PROP, was introduced in [5] that achieved large improvements over traditional "deterministic" iterative-improvement techniques like FM [7] and LA [10]. While PROP's gain function has a greater futuristic component than FM or LA, it incorporates spatially local information—only information on the removal probabilities of adjacent nets of a cell is used in its gain computation. This prevents a higher-level view of non-local structures. Also, giving uniform weights to all nets, results in an inability to differentiate between the futuristic benefit of removing one net from another. In this paper, we present a more sophisticated partitioner DEEP-PROP that incorporates more non-local (second-order) structural information than PROP. The second-order information is incorporated into cell gains as well as variable net weights—the latter helps to focus future cell moves in a cluster around the currently moved cell and thus better utilizes the information that led to its selection. A lower-complexity version, VAR-PROP, that also uses dynamically assigned variable net weights, but based on first-order information, has also been developed. Both versions yield significant improvements over PROP on the ACM/SIGDA benchmark suite. DEEP-PROP yields mincut improvements of as much as 39% for large circuits and an average improvement of 20% over all circuits; it is about 3.8 times slower than PROP, which is very fast. VAR-PROP, which has a much lower computational complexity than DEEP-PROP, yields maximum and average mincut improvements over PROP of 27% and 12%, respectively, while being only about 1.14 times slower.

1. Introduction

VLSI circuit partitioning is an important processing step for many applications in VLSI design such as testing, placement and multiple-FPGA implementation. While many different partitioning methods have been proposed in the past, iterative-improvement based approaches are widely used due to their efficiency and flexibility in adapting to different optimization objectives. The classical iterative-improvement methods include those of Kernighan-Lin (KL) [9] and Fidducia-Mattheyses (FM) [7]; the latter is a lower-complexity version of the KL algorithm that considers single cell moves instead of cell-pair swaps. The techniques work well for small to medium-sized circuits with up to ten thousand cells, but rapidly break down in terms of solution quality for larger circuits due to their "greedy" gain functions. More recently, a number of more powerful iterative-improvement based partitioners like PANZA [12], GMetis [2], STRAWMAN [8], PROP [5], CLIP/CDIP [6], hMetis [11], ML$_c$ [1] and LSR/MFFS [3] have been developed, that yield very good results. However, there still are conceptual issues that warrant further study in order to yield more insights and improvements in partitioning algorithms, thus enabling them to deal more effectively with high-complexity circuits. This paper studies one such issue, the computation and use of stochastic and non-local information in cell gain functions that drive iterative-improvement partitioners.

Our main goal in this work is not necessarily to produce the next best partitioning program, but to delve into the concepts of stochastic and non-local structural information, and their focused use, and to demonstrate that by using powerful gain functions, even flat partitioners can obtain results comparable to multilevel ones. DEEP-PROP's results are within 8% of the current best partitioners, and can very likely be improved further by using orthogonal paradigms like multilevel partitioning. However, flat partitioners are important, especially in complex placement systems (e.g., timing-driven placement in the presence of multiple constraints), where it may be detrimental to "hide" useful information about the circuit by clustering subcircuits into large "nodes", as is done in multilevel partitioners.

The rest of the paper is organized as follows. In Sec. 2., we recap the gain functions of other relevant methods, FM, LA and PROP. Section 3. introduces the notion of second-order information based on removal probabilities of level-2 nets, and the advantages of incorporating this information. In Sec. 4., we present techniques for incorporating second-order information as well as schemes for net weight determination based on this information. Section 5. formally presents the algorithm and derives time complexities. Sec. 6. presents our experimental results, and we conclude in Sec. 7..

2. Relevant Past Methods

The mincut bi-partitioning problem can be formally stated as follows. A circuit is represented by a hypergraph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes that represent *cells* or *modules* in the circuit, and $E = \{n_1, n_2, ..., n_e\}$ is the set of hyperedges which represent *nets* in the circuit; n and e are the total numbers of cells and nets in G, respectively. Each net n_i will be represented as a subset of the cells that it connects, and $E(u)$ is the set of nets connected to cell u. The *area* of each cell is denoted as $a(v_i)$. In a bi-partitioning problem, V is partitioned into two subsets, V_1 and V_2, such that each v_i belongs to either V_1 or V_2. We denote the set of nets with incident cells in both V_1 and V_2 by E_{cut}, and the total size of V_1, V_2 by $|V_1|$, $|V_2|$ respectively, i.e., $|V_j| = \sum_{v_i \in V_j} a(v_i)$ for j = 1, 2.

There are one or more objectives and constraints that any parti-

[*]This work was supported partly by an UIC Research Grant

tioner must achieve and comply with, respectively. One of the most common objective is to minimize the cutsize of the cutset which is defined as $cutsize = \sum_{n_i \in E_{cut}} c(n_i)$. where $c(n_i)$ is the $cost$ or $weight$ of net associated with n_i. Based on this objective, every cell u in the circuit can be assigned a gain $gain(u)$, which is an estimate of the improvement of the objective engendered by moving u to the other subset. The gain function is the most important determinant of the efficacy of an iterative-improvement algorithm, and can range from simplistic to complex. The gain function is defined differently for different partitioners like FM, LA, and PROP. Both FM and LA define a cell's gain (LA has a gain vector) as a deterministic one:

$$gain(u)[k] = \sum_{n_i \in E_k(u)} c(n_i) - \sum_{n_j \in I_k(u)} c(n_j),\ k = 1 \text{ to } l$$

where $E_k(u)$ is the set of nets that have exactly k cells (including u) in u's subset, and $I_k(u)$ is the set of nets that have exactly $k-1$ cells in the other subset [10], [7]; l is the degree of "look-ahead", and is 1 for FM which lacks of any look-ahead information. PROP's gain function captures more accurate futuristic information (thus yielding much better results [5]) as follows.

PROP [5] defines a probabilistic gain for every cell u and its adjacent nets based on the cells' initially estimated approximate probabilities. A cell u's probability $p(u)$ is then computed from its gain using a monotonically increasing function $f(g(u))$ (usually a linear function). The probabilistic gain of cell u, assuming $u \in V_1$, is given by

$$gain(u) = \sum_{n_i \in E(u)} g_{n_i}(u) \quad (1)$$

where each gain component $g_{n_i}(u)$ for net n_i is formulated as

$$g_{n_i}(u) = c(n_i)[(\text{Prob. of moving } n_i \text{ to } V_2 \text{ given } u'\text{s move}) -$$

$$(\text{Probability of moving } n_i \text{ to } V_1 \text{ given that } u \text{ is not moved})]$$

The rationale for the negative term in the above expression is that moving u precludes the event of moving n_i to V_1 from occurring, and thus eliminates the possibility of removing n_i from the cutset in that manner. Using conditional probabilities, this leads to the expression

$$g_{n_i}(u) = c(n_i)[\prod_{u_x \in n_{i,1}} p(u_x))/p(u) - \prod_{u_y \in n_{i,2}} p(u_y)] \quad (2)$$

where $n_{i,j} = n_i \cap V_j$, for $j = 1, 2$. Note that the above expression also applies to nets n_i that are not in the cutset, but lie entirely in the subset that u is in, viz., V_1 for the above expressions. In this case, the probability of moving n_i to V_1 (i.e., the second term in Eqn. 2) is 1, and thus $g_{n_i}(u)$ is a penalty (i.e., it is negative) equal to the probability of not being able to move n_i out of the cutset after it is introduced in it by moving u from V_1 to V_2.

To begin the partitioning process, PROP refines the cells' probabilistic gains and probabilities with one or two iterations of probability assignment and gain calculation. After moving the current best cell u at each step (assuming no constraints are violated), PROP records the immediate gain of u and at the end of each pass computes the maximum prefix of immediate gains. While u's probabilistic gain is an indication of future immediate gains from moving other nodes, and is thus a good node selection function, the immediate gain needs to be recorded for each move to compute the maximum improvement point.

As reflected in PROP's cell gain function expressed in Eqn. 1 and Eqn. 2, PROP captures information on the advantage of moving a cell based only on the removal probabilities of its incident nets—these are called *level*-1 nets, and information based on them is referred to as *first-order information*. Besides 1st-order information, DEEP-PROP also incorporates in the cell gain function, the futuristic benefit of removing each incident net n_i from the cutset in terms of the subsequent (after n_i is removed) removal probabilities of nets adjacent to n_i; a net n_j is said to be *adjacent* to another net n_i if there is at least one common cell v between them. Nets adjacent to level-1 nets of a cell u that are themselves not its level-1 nets, are called *level*-2 nets with respect to u. The removal probabilities of level-2 nets, called *second-order information*, represents non-local structures around each cell u, and is incorporated in its gain function.

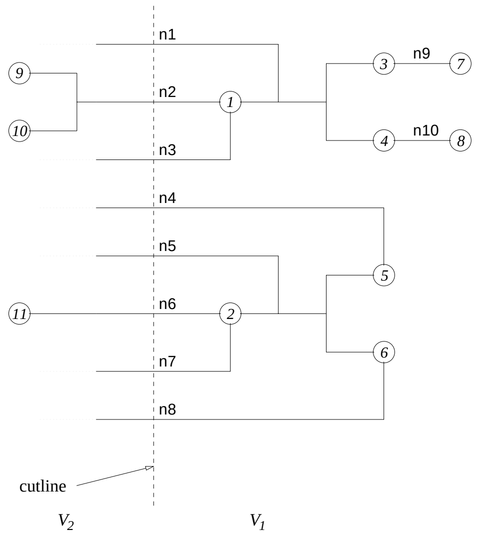

Figure 1: Base cell selection which shows the advantage of using the second-order information.

Another difference between PROP and DEEP-PROP is that the former treats all nets in the circuit uniformly in terms of their significance, whereas DEEP-PROP gives nets different *weights* (or *priority factors*). The priority factor for a net n_i is based on much its removal from the cutset increases or decreases the probabilities of either removing nets adjacent to it from the cutset or maintaining them as internal nets on a particular subset. Higher priority factors translate to higher cell gains for those cells that promote the removal of corresponding nets from the cutset, and lower cell gains for those that inhibit this removal (e.g., by locking a net in the cutset). This enables DEEP-PROP to better identify nets and their connected cells that yield higher benefits by being removed from the cutset.

3. Using Second-Order Information

3.1. Advantages of Second-order Information

Figure 1 illustrates the benefit of using second-order information. For simplicity, we assume that all cells shown in the figure have the same move probabilities, and all nets in the cutset have the same probabilities of being removed from it by cell moves from V_1 to V_2 (this is also called the *going probability* to V_2). As far as PROP is concerned, the difference between the gains of cells 1 and 2 are only due to the components corresponding to net n_2 and n_6, respectively. Hence, PROP will compute a larger gain for cell 1. However, closer scrutiny reveals that cell 2 is a much better candidate to move because the move of cell 2 engenders the removal of level-1 net n_5 (among other level-1 nets), which in turn engenders the removal of level-2 nets n_4 and n_8. The removal of the corre-

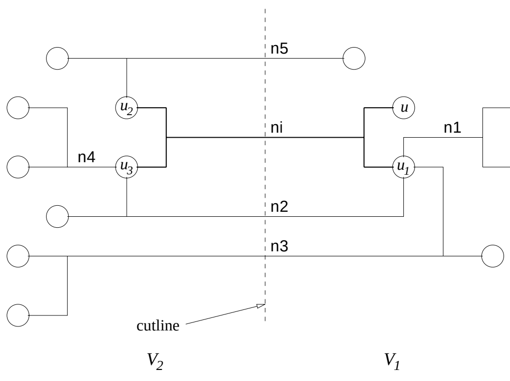

Figure 2: Example circuit used for determining the benefit of removing net n_i based on information provided by its neighboring nets n_1, n_2, n_3 n_4 and n_5.

sponding net n_1 by moving cell 1, on the other hand, does not facilitate the removal of any level-2 nets from the cutset; on the contrary, it leads to the introduction of level-2 nets n_9 and n_{10} into the cutset. Since PROP does not take second-order information into account, it cannot distinguish between the benefits of removing net n_5 versus removing net n_1—these benefits are not reflected on the gains of their incident cells 2 and 1, respectively. Thus it is clear that by considering level-2 nets, the gain function can be made more accurate.

In order to incorporate 2nd-order information in our new techniques, the *benefit factor (BF)* of each net will be computed—it indicates the benefit of removing it from the cutset in terms of the subsequent removal probabilities of adjacent cut-nets (nets in the cutset), which contributes positively to the BF, and the cutset insertion probabilities of adjacent internal nets (nets not in the cutset), which contributes negatively to the BF. Hence, for the circuit shown in Fig. 1, net n_1 will be given a much lower benefit factor than net n_5.

3.2. Formulation

We denote by $C_s(n_x, n_y)$ the set of cells in partition subset s that the two nets n_x and n_y have in common, and $C_{\overline{s}}(n_x, n_y)$ is its counterpart for the other subset $\overline{s}$ of the partition. Hence, $C_1(n_i, n_1) = \{u_1\}$ and $C_2(n_i, n_1) = \emptyset$ for the circuit in Fig. 2. Further, we denote $N_s(n_x)$ to be the set of neighboring nets n_y of n_x for which $C_s(n_x, n_y) \neq \emptyset$. For the circuit in Fig. 2, $N_1(n_i) = \{n_1, n_2, n_3\}$ and $N_2(n_i) = \{n_2, n_4, n_5\}$. Note that, $N_s(n_x) \cup N_{\overline{s}}(n_x) = N(n_x)$, the set all nets adjacent to n_x, and $C_s(n_x, n_y) \cup C_{\overline{s}}(n_x, n_y) = C(n_x, n_y)$, the set of all common cells between n_x and n_y. We introduce a new term $G(n_i)$ which is the *2nd-order benefit factor* of net n_i. For nets that are in the cutset, there are two components of $G(n_i)$, namely $G(n_i^{s\to\overline{s}})$ and $G(n_i^{\overline{s}\to s})$. $G(n_i^{s\to\overline{s}})$ is the *expected benefit*, given n_i is removed from the cutset by moving all its incident cells from subset s to subset $\overline{s}$, whereas $G(n_i^{\overline{s}\to s})$ is the expected benefit for moving n_i from $\overline{s}$ to s. Hence, for nets that are not in the cutset, there is only one component of $G(n_i)$, $G(n_i^{s\to\overline{s}})$ or $G(n_i^{\overline{s}\to s})$, depending on whether n_i is internal to subset s or subset $\overline{s}$, respectively. The value of $G(n_i)$ is determined by the neighboring nets of n_i as follows.

$$G(n_i^{s\to\overline{s}}) = \sum_{n_j \in N_s(n_i)} G_{n_j}(n_i^{s\to\overline{s}}) \tag{3}$$

$$G(n_i^{\overline{s}\to s}) = \sum_{n_j \in N_{\overline{s}}(n_i)} G_{n_j}(n_i^{\overline{s}\to s}) \tag{4}$$

where for a net n_j in the cutset,

$$G_{n_j}(n_i^{s\to\overline{s}}) = c(n_j)[\text{Probability of removing } n_j \text{ from the cutset given that } n_i \text{ is moved from } s \text{ to } \overline{s}]$$

Thus

$$G_{n_j}(n_i^{s\to\overline{s}}) = c(n_j) \cdot p(n_j^{s\to\overline{s}}) / \prod_{u \in C_s(n_i, n_j)} p(u) \tag{5}$$

where $p(n_j^{s\to\overline{s}})$ is the probability of net n_j being removed from the cutset by moving all the adjacent cells from s to $\overline{s}$ (also called the *going probability* of n_j from s), and using the derivation in [5]

$$p(n_j^{s\to\overline{s}}) = \prod_{v \in n_i \cap s} p(v)$$

For a net n_j internal to subset s,

$$G_{n_j}(n_i^{s\to\overline{s}}) = -c(n_j)[\text{Prob. that } n_j \text{ will remain in the cutset given that it is introduced in it by moving } n_i \text{ from } s \text{ to } \overline{s}]$$

Thus

$$G_{n_j}(n_i^{s\to\overline{s}}) = -c(n_j) \cdot (1 - p(n_j^{s\to\overline{s}})) / \prod_{u \in C_s(n_i, n_j)} p(u) \tag{6}$$

A similar set of equations holds for moves from $\overline{s}$ to s. Note that $G(n_i^{s\to\overline{s}})$ and $G(n_i^{\overline{s}\to s})$ can be either positive or negative.

Using this second-order information given by Eqns. 3 to 6, DEEP-PROP uses the following gain functions $g_{n_i}(u)$ for a cell $u \in V_1$. For a net n_i in the cutset

$$g_{n_i}(u) = [(\text{Expected 1st \& 2nd order benefit of moving } n_i \text{ from } V_1 \text{ to } V_2) \times (\text{Probability of moving } n_i \text{ to } V_2 \text{ given that } u \text{ has been moved})] - [(\text{Expected 1st \& 2nd order benefit of moving } n_i \text{ from } V_2 \text{ to } V_1) \times (\text{Probability of moving } n_i \text{ to } V_1 \text{ given that } u \text{ is not moved})]$$

Similar to PROP's gain function, the rationale for the negative term in the above expression is that moving u precludes the event of moving n_i to V_1 from occurring, and thus eliminates the corresponding expected benefits of n_i. Also note that the 1st-order benefit of removing n_i from the cutset is simply $c(n_i)$, and is used in PROP's gain function (Eqn. 2). We thus obtain

$$g_{n_i}(u) = [G(n_i^{1\to 2}) + c(n_i)]p(n_i^{1\to 2})/p(u) - [(G(n_i^{2\to 1}) + c(n_i))]p(n_i^{2\to 1}) \tag{7}$$

Substituting for $p(n_i^{1\to 2})$ and $p(n_i^{2\to 1})$,

$$g_{n_i}(u) = [\sum_{n_j \in N_1(n_i)} G_{n_j}(n_i^{1\to 2}) + c(n_i)][(\prod_{u_x \in n_{i,1}} p(u_x))/p(u)] - [\sum_{n_j \in N_2(n_i)} G_{n_j}(n_i^{2\to 1}) + c(n_i)][\prod_{u_y \in n_{i,2}} p(u_y)] \tag{8}$$

For net n_i internal to V_1, the above formulation of $g_{n_i}(u)$ still holds, except that in this case $G(n_i^{2\to 1}) = 0$. Hence

$$g_{n_i}(u) = -(c(n_i) - [G(n_i^{1\to 2}) + c(n_i)][p(n_i^{1\to 2})/p(u)]) \tag{9}$$

or

$$g_{n_i}(u) = -(c(n_i) - [\sum_{n_j \in N_1(n_i)} G_{n_j}(n_i^{1\to 2}) + c(n_i)] \times [(\prod_{u_x \in n_{i,1}} p(u_x))/p(u)]) \tag{10}$$

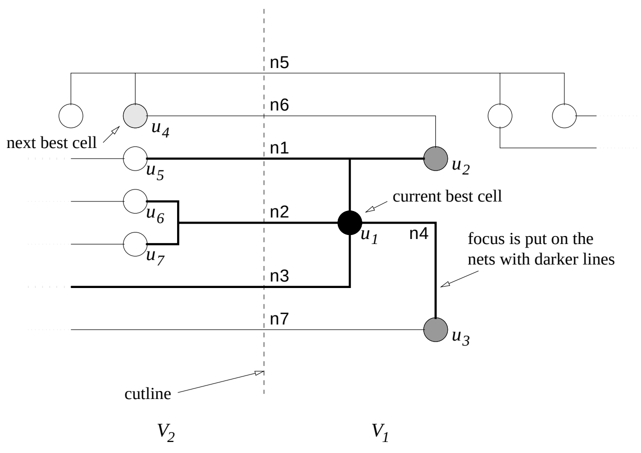

Figure 3: The focus on removing level-1 nets n_1, n_2, n_3 and n_4, and level-2 nets n_6 and n_7 should be maintained after moving u_1 by giving more weight to the level-1 nets.

4. Variable Net Weights

4.1. Determining Net Weights

Based on DEEP-PROP's gain function, let u_1 be the highest-gain cell moved; u_1 is selected because it will engender the movement of the most number of "removable" (nets that can be removed from the cutset with high probability) level-1 and level-2 nets. This initial effort should not be stopped after the move of cell u_1. Otherwise, the information regarding removable level-1 and level-2 nets contained in $g(u_1)$ will be wasted, especially if the next best cell(s) w moved are in the opposite direction to u_1's move. In other words, the focus of the partitioner in removing u_1's level-1 and level-2 nets shifts to removing w's level-1 and level-2 nets, and may shift again in the next moves. Thus there is a potential problem of the partitioner moving from one promising cluster to another without pulling an entire cluster out of the cutset; this can and does result in many removable clusters being locked in the cutset. The problem is illustrated in Fig. 3. In the current situation, cell u_1 is the best cell to move and cell u_4 is the second best. Moving cell u_1 is an initial effort towards removing level-1 nets n_1, n_2, n_3, n_4 as well as level-2 nets n_6 and n_7 by cell moves from V_1 to V_2. However, since there is no direct connection between cells u_1 and u_4, the move of u_1 will not alter the gain of the next-best cell u_4. Hence cell u_4 will be moved next, and this reduces the gain of cell u_2, thus reducing the probability of removing nets n_1 and n_6 from V_1 to V_2.

After moving the best cell u_1, the goal should thus be to follow through with cell moves that engender the removal of high-benefit level-1 and level-2 nets that contributed to the selection of u_1. The level-1 cut-nets of u_1 are n_1, n_2, and n_3 for the circuit in Fig. 3. The level-2 nets will be automatically removed either fully or partially once the level-1 nets are removed. The partitioner's focus on these nets can be retained by simply amplifying the weight of level-1 nets of the moved cell by a constant factor. This will amplify either positively or negatively the gains of free cells which are connected to these level-1 nets (i.e., neighboring cells of u_1). This will promote those neighboring cells which are on the same (original) subset as u_1, and will demote those on the other subset. Therefore, cells u_2 and u_3 will have positive amplification while cells u_5, u_6 and u_7 will have negative amplification from their corresponding nets. The move of u_1 will then be followed by cells u_2 and u_3 either immediately after or in near-future moves. It can thus be ensured that nets that are intended to be removed (stemming from the decision to move cell u_1) do indeed get removed, and that the partitioner retains focus on a particular subset of nets before it shifts its effort to others.

However, the constant factor amplification will not always work well, since different nets have different benefit factors—a more refined method is needed. If all level-1 nets of a moved node are given a uniformly high weight, then even those nets and neighbors that have very low probabilities of being moved will accumulate high gains that are misleading. This will cause the partitioner to focus too heavily on removing such nets with negative consequences (e.g., those with low or negative expected benefits—meaning that they cause other nets to get introduced or locked in the cutset), and at the expense of diverting attention from other more promising nets. Rather than giving a constant weight to every net, they are given different *priority factors* $\Phi(n_i)$'s according to their degree of "goodness" of being removed from the cutset. The goodness of a net n_i can be measured in two ways: one is based on the expected benefit $G(n_i^{1\to 2})$ (used by DEEP-PROP), and the other one is based on the going probability of n_i from a subset, say V_1 ($p(n_i^{1\to 2})$). The version which uses the second variable net-weight scheme is called VAR-PROP. Assigning net weights in this manner allows the partitioner to properly reward or penalize nets and their connected cells. This serves the purposes of focusing on nets to be removed as well as prioritizing these nets according to their expected benefits (nets whose removal will likely lead to the removal of other adjacent nets should be focused on first) or their removal probabilities (nets that are easier to remove should be focused on first).

4.2. Formulation

The priority factor $\Phi(n_i)$ on a net n_i is normalized with a linearly increasing function given by

$$\Phi(n_i) = \begin{cases} \Phi_{max} & \text{when } \Lambda(n_i) > T_{up} \\ \Phi_{min} + (\Phi_{max} - \Phi_{min})\times & \text{when} \\ (\Lambda(n_i) - T_{low})/(T_{up} - T_{low}) & T_{low} \le \Lambda(n_i) \le T_{up} \\ \Phi_{min} & \text{when } \Lambda(n_i) < T_{low} \end{cases}$$

where $\Lambda(n_i)$ is the metric of goodness being used ($G(n_i^{s\to\overline{s}})$ for DEEP-PROP, and $p(n_i^{s\to\overline{s}})$ for VAR-PROP), Φ_{max}, Φ_{min}, T_{up}, and T_{low} limit the upper and lower bounds of $\Phi(n_i)$ and $\Lambda(n_i)$. Hence the higher is $G(n_i^{s\to\overline{s}})$ or $p(n_i^{s\to\overline{s}})$, the higher is its $\Phi(n_i)$. Let u_1 be the current best cell being moved from V_1 to V_2; when cell u_1 is moved for the first time on a net n_i, for all unlocked neighbors u of u_1 on n_i, $g_{n_i}(u) = \Phi(n_i) g_{n_i}^{upd}(u)$, where $g_{n_i}^{upd}(u)$ is the $g_{n_i}(u)$ after normal PROP-type updation of neighbors of u_1 after it is moved—see algorithm DEEP-PROP in Fig. 4 for details. Further, $g_{n_i}(u) \ge 0$ for all $u \in n_{i,1}$ whereas $g_{n_i}(u) \le 0$ for all $u \in n_{i,2}$; recall that $n_{i,s} = n_i \cap V_s$, $s = 1, 2$. This means that all neighbors of u_1 in V_1 will have positive magnification from the corresponding nets; in contrast, all the neighbors in V_2 will have negative magnification.

To illustrate this further, let us consider the example circuit of Fig. 3, and apply the variable net-weight method used by DEEP-PROP after cell u_1 is moved from V_1 to V_2. The following conditions hold after the move: $E(u_1) = \{n_1, n_2, n_3, n_4\}$, $G(n_1^{1\to 2}) = 1$, $G(n_2^{1\to 2}) = 0$, $G(n_3^{1\to 2}) = 0$, and $G(n_4^{1\to 2}) = 1$. Thus the gain of cells u_2 and u_3 due to nets n_1 and n_4 will be positively magnified, while the gains of cells u_5, u_6, and u_7 due to nets n_1 and n_2 will be negatively magnified. A similar process applies to VAR-PROP which uses $p(n_1^{1\to 2})$, $p(n_2^{1\to 2})$, $p(n_3^{1\to 2})$, and $p(n_4^{1\to 2})$ to determine the $\Phi(n_i)$ values.

In the actual implementation, all cell gains are initially shrunk by a factor σ, $0 \le \sigma \le 1$, to yield the initial cell gains, i.e., initially, $g(u) = \sigma \cdot g(u)$. However, the un-shrunk gains and all their components values are stored. All the update procedures after the initial

```
**Procedure** DEEP-PROP($G$)
/* $G$ is the hypergraph to be partitioned */
**Repeat**
  1. Assign each cell an uniform initial probability $p_{init}$ of, say, 0.5.
  2. Compute the initial expected benefits of all nets, and the gains of all
     cells according to Eqn. 3 to Eqn. 10.
  3. Recompute cell probabilities $p(u)$'s using a monotonic function
     $f(g(u))$ of $g(u)$'s.
  4. Sort all the cells based on their gain separately for $V_1$ and $V_2$.
  5. Shrink all the cell gains by a common factor $\sigma < 1$ ($g(u) = \sigma g(u)$).
  6. **Repeat** Steps 7 to 9 **until** all cells are moved.
  7. Choose the base cell $u$ based on the cell's updated gain and the bal-
     ance criterion. Move the cell, update its neighbors.
  8. **For** each $n_i \in E(u)$ update $G_{n_i}(n_j)$ for all $n_j \in N(n_i)$.
  9. **For** each unlocked neighboring cells $w$ on each net $n_i \in E(u)$, **do**
     Steps (a) to (b).
       (a) Update $g_{n_i}(w)$ with un-shrunk gain components and compute
           the $\Phi(n_i)$ for nets $n_i$.
       (b) $g_{n_i}(w) = \Phi(n_i) \cdot g_{n_i}(w)$.
  10. Find the point in the move sequence which corresponds to the mini-
      mum cutsize, and reverse all the moves after this point.
**Until**(the min. cutsize obtained in this *pass* [Steps 1-10] does not represent
any improvement over the previous pass)
```

Figure 4: The second-order probability-based partitioning algorithm DEEP-PROP

move use the un-shrunk values and the calculated $\Phi(n_i)$'s. This results in giving net n_i a weight factor of $\Phi(n_i)/\sigma$ after a node connected to it has been moved for the first time. The next section formally presents algorithm DEEP-PROP, which incorporates second-order information and variable net weights into a probabilistic gain function.

5. Algorithm Description and Complexities

Figure 4 formally describes the partitioning algorithm DEEP-PROP. The flow of the partitioning process is essentially the same as PROP except for some additional steps during the initial cell gain calculation and updation. An initial probability is assigned to every cell at the start. Based on the cell probabilities and nets' expected benefits, the cell probabilistic gains are computed using Eqn. 3 to Eqn. 10. All cell gains and their components are then shrunk by a factor of $\sigma < 1$; the cell ordering remains unchanged. The max-gain cell is selected, moved to the other subset and locked. The expected benefits of all adjacent nets (level-1 nets) are first updated, then gains of cells connected to them are updated with the corresponding $\Phi(n_i)$ values. Note that the updating process uses the un-shrunk gain components $g_{n_i}(u)$s. The next best cell is then chosen and the process continues until all cells are locked.

Let ρ_n, ρ_u be the numbers of adjacent cells, nets (pins) on net n and cell u respectively, and ρ^{net}_{max}, ρ^{cell}_{max} be the maximum ρ_n, ρ_u over all nets and cells, respectively, in the circuit. The number of nets whose benefit factors $G(n_i)$ need to be updated for each move is $O(\rho^{net}_{max}\rho^{cell}_{max})$. Hence in the worst case, the timing complexity of DEEP-PROP is $O(\rho^{net}_{max}\rho^{cell}_{max})$ per cell move, or $O(n\rho^{net}_{max}\rho^{cell}_{max})$ over the entire pass. The $\rho^{net}_{max}\rho^{cell}_{max}$ value is not constant throughout a pass; it decreases during a pass as cells and nets are locked and their gains and benefit factors, respectively, do not need to be updated. Thus the time complexity is much lower in practice.

6. Experimental Results

DEEP-PROP was implemented in C++ on top of PROP. Experiments have been performed on the ACM/SIGDA benchmark suite. The characteristics of these circuits are given in Table 1. All experimental results presented in Table 1 (comparisons of DEEP-PROP and VAR-PROP with PROP) were run on a SPARC-Station 20. The relevant common parameters values for both DEEP-PROP and VAR-PROP are: initial probability = 0.3, $\sigma = 0.1$, $\Phi_{max} = 2$, $\Phi_{min} = 1$. For DEEP-PROP, $T_{up} = 8$, $T_{low} = -8$, and for VAR-PROP $T_{up} = 0.9$, $T_{low} = 0.1$.

Table 1 shows that most of the improvements yielded by DEEP-PROP are for large-size circuits (near the bottom of the table). For the largest circuit `golem3` the improvement is 15.6% and 26.8% in mincut and average-cut, respectively, over both PROP with 20 and 40 runs. Also, there is an improvement of 39% in mincut for `avq_large` over both runs of PROP. Circuit `s38417` improves in mincut by 38% and 19% over PROP 20 and 40 runs, respectively. Mincut improvement is also obtained for `avq_small` which is as much as 15% over both PROP with 20 and 40 runs. Perhaps the most obvious and consistent improvement is in the average-cut, where for all circuits except `t2` there are significant improvements. This means that for a particular run, DEEP-PROP's final result has less dependency on the initial random partition compared to PROP. For small to medium size circuits, DEEP-PROP is about 2 times (or less) slower than PROP. Large-size circuits such as `avq_large` and `avq_small` have many nets with large net-degrees which increase the computational time significantly during the updating procedures. Over all circuits, DEEP-PROP shows mincut improvements of 13.2% over PROP 20-runs and 9% over PROP 40-runs, and it is about 3.8 times slower than PROP (with 20 runs), which is quite fast. Comparisons between PROP and VAR-PROP are also shown in the same table, where it is more appropriate to compare VAR-PROP to PROP with 20 runs than 40 runs, since VAR-PROP is comparable in speed with PROP 20-runs (see last 2 rows of Table 1). Similar to DEEP-PROP, improvement in mincut are most noticeable in larger circuits such as `golem3`, `avq_large` and `avq_small`—these improvements are by 11.5%, 27% and 20%, respectively over both PROP 20 and 40 runs. Again, improvements in the average-cut are very noticeable for all the circuits except `t2`. Even though the results of DEEP-PROP are slightly superior to VAR-PROP, the speed of VAR-PROP makes it a very attractive partitioner. VAR-PROP runs almost as fast as PROP because it has no extra data structures, and its additional time complexity is constant with respect to problem size. Over all circuits, VAR-PROP improves over PROP in both mincut and average-cut by 11.9%, while being only 1.14 times slower than PROP. However, if a very high-quality mincut result is desired, DEEP-PROP should be the partitioner of choice, especially for large circuits.

7. Conclusions

We introduced the concept of stochastic second-order structural information, in terms of the expected benefits of removing connected (level-1) nets from the cutset, in the cell gain function. In order to use this information in a focused manner, we also developed a refined net weight amplification technique for nets on which a cell has been moved for the first time. These concepts resulted in two new stochastic-gain based partitioners DEEP-PROP and VAR-PROP. Both partitioners yield much better mincut results than PROP [5], which uses only first-order structural information and uniform non-amplified net weights in its (stochastic) gain function. On the suite of ACM/SIGDA benchmark circuits, DEEP-PROP yields mincut improvements of as much as 39% for large circuits and an average improvement of 20% over all circuits in the above suite; it is about 3.8 times slower than PROP, which is very fast. VAR-PROP, which has a much lower computational complexity than DEEP-PROP, yields maximum and average mincut improvements over PROP of 27% and 12%, respectively, while being only about 1.14 times slower.

Some of the current state-of-the-art partioners are mostly of

Test Case Characteristics				PROP 20-runs			PROP 40-runs			DEEP-PROP 20-runs			VAR-PROP 20-runs		
Name	**# Cells**	**# Nets**	**# Pins**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**	**Min**	**Avg**	**Time**
balu	801	735	2697	27	33.2	0.75	27	33.9	0.70	27	30.9	1.18	27	29.9	0.69
p1	833	902	2908	47	60.6	0.86	47	61.0	0.88	47	58.4	1.48	47	56.9	0.97
bm1	882	903	2910	47	59.8	0.97	47	60.2	1.00	47	58.1	1.55	47	57.5	1.00
t4	1515	1658	5975	56	70.2	2.12	51	68.8	2.08	50	59.5	6.00	49	61.4	2.46
t3	1607	1618	5807	58	72.6	2.47	58	72.2	2.35	57	67.4	5.20	58	66.0	2.26
t2	1663	1720	6134	91	100.2	2.99	89	99.5	2.82	88	101.0	5.59	88	99.3	2.89
t6	1752	1641	6638	68	85.9	3.64	68	88.2	3.48	60	74.0	6.50	64	75.5	3.76
struct	1952	1920	5471	36	49.1	1.90	33	46.1	1.85	33	40.3	2.28	33	42.7	1.70
t5	2595	2750	10076	83	98.7	4.71	76	98.2	4.75	75	92.0	11.53	72	86.6	4.85
19ks	2844	3282	10547	107	140.6	4.49	107	138.7	4.70	106	121.3	12.16	107	121.7	4.31
p2	3014	3029	11219	145	189.0	6.43	145	191.1	6.96	141	186.4	14.43	143	186.3	8.42
s9234	5866	5844	14065	45	62.1	6.80	45	69.4	7.04	47	59.0	8.56	45	63.2	6.04
biomed	6514	5742	21040	84	116.5	15.56	84	112.3	15.95	83	107.0	24.39	84	101.2	14.36
s13207	8772	8651	20606	74	111.2	10.98	74	109.7	11.06	71	91.4	18.17	71	101.5	11.16
s15850	10470	10383	24712	66	99.2	17.80	50	100.8	18.05	62	90.7	21.31	72	96.3	16.73
industry2	12637	13419	48404	223	321.2	45.56	200	304.1	48.91	201	285.1	147.25	183	270.4	66.76
industry3	15406	21924	68290	278	360.1	45.95	260	359.4	48.97	243	305.6	141.46	243	320.2	65.56
s35932	18148	17828	48145	73	80.9	30.70	73	82.5	30.28	73	73.7	51.76	73	77.3	27.25
s38584	20995	20717	55203	74	103.5	63.71	62	101.3	59.78	63	99.3	74.37	52	110.5	59.95
avq_small	21918	22124	76231	208	416.1	45.09	208	415.7	46.63	177	304.8	469.74	167	305.4	53.79
s38417	23949	23843	57613	95	129.2	50.21	73	125.9	51.06	59	104.2	61.79	75	114.2	51.81
avq_large	25178	25384	82751	337	441.4	54.87	337	461.6	52.54	205	368.6	514.22	245	377.2	70.22
Total				2322	3201.3	418.56 x20 = 8371.2	2214	3200.6	421.84 x40 = 16873.6	2015	2778.7	1600.92 x20 = 32018.4	2045	2821.2	476.94 x20 = 9538.8
% Improvement over PROP 20-runs				-	-	-	4.65	0.02	-101.57	13.22	13.20	-282.48	11.93	11.87	-13.95
golem3	103048	144949	338419	1681	2232.3	340.27 x20 = 6805.4	1681	2266.1	347.58 x40 = 13903.2	1419	1634.5	1352.23 x20 = 27044.6	1487	2040.6	871.30 x20 = 17426
% Improvement for golem3				-	-	-	0	-1.52	-104.31	15.59	26.79	-297.41	11.54	8.6	-156.08

Table 1: Benchmark circuit characteristics and comparisons of cutsizes for the $45 - 55\%$ balance criterion produced by PROP-20 runs, PROP-40 runs, and DEEP-PROP and VAR-PROP, each with 20 runs. The **Min** column is the best result from all the runs, the **Avg** column shows the average cutsize over those runs, and the **Time** column is the CPU time (in seconds) per run.

the iterative-improvement variety, and include CLIP/CDIP [6], PANZA [12], STRAWMAN [8], hMetis [11], ML$_C$ [1], and LSR/MFFS [3]. We do not explicitly compare our results to these techniques, except to note that DEEP-PROP's results are within 8% of the best of these, and that its results can most likely be improved further by using orthogonal paradigms like multilevel partitioning used in many of the above partitioners. However, our goal here was not necessarily to produce the next best partitioning program, but to delve into the concepts of stochastic and non-local structural information and their focused use, and to demonstrate that by using powerful gain functions, even flat partitioners can obtain results comparable to multilevel ones. From a bigger picture standpoint than just obtaining the best mincut results, flat partitioners are important, especially in complex placement systems (e.g., timing-driven placement in the presence of multiple constraints), where it may be detrimental to "hide" useful information about the circuit by clustering subcircuits into large "nodes", as is done in multilevel partitioners.

References

[1] C.J. Alpert, J-H. Huang and A.B. Kahng, "Multilevel circuit partitioning", *Proc. Design Automation Conf.*, June 1997, pp. 530-533.

[2] C.J. Alpert and A.B. Kahng, "A hybrid multilevel/genetic approach for circuit partitioning", *Physical Design Workshop*, 1996, pp. 100-105.

[3] J. Cong, et al., "Large scale circuit partitioning with loose/stable net removal and signal flow based clustering", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, Nov. 1997, pp. 441-446.

[4] S. Dutt and H. Theny, "Partitioning Around Roadblocks: Tackling Constraints with Intermediate Relaxations", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, IEEE/ACM ICCAD, Nov., 1997, pp. 349-355.

[5] S. Dutt and W. Deng, "A probability-based approach to VLSI circuit partitioning", *Proc. Design Automation Conf.*, June 1996, pp. 100-105.

[6] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, Nov. 1996, pp. 92-99.

[7] C.M. Fiducia and R.M. Mattheyses, "A linear-time heuristic for improving network partitions", *Proc. Nineteenth Design Automation Conf.*, 1982, pp. 175-181.

[8] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techniques", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 8, pp. 849-866, August 1997.

[9] B.W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", *Bell System Tech. Journal*, vol. 49, February 1970, pp. 291-307.

[10] B. Krishnamurthy, "An improved mincut algorithm for partitioning VLSI networks", *IEEE Trans. on Computers*, vol. C-33, no. 5, May 1984, pp. 438-446.

[11] G. Karypis, et al., "Multilevel hypergraph partitioning: Application in VLSI domain", *Proc. Design Automation Conf.*, June 1997, pp. 526-529.

[12] J. Li, J. Lillis and C-K. Cheng, "Linear decomposition algorithm for VLSI design applications", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, 1995, pp. 223-228.