Partitioning Around Roadblocks:
Tackling Constraints with Intermediate Relaxations

Shantanu Dutt
dutt@eecs.uic.edu
Dept. of Electrical Eng. and Comp. Science
Univ. of Illinois at Chicago
Minneapolis, MN 55455, USA

Halim Theny
theny@ee.umn.edu
Dept. of Electrical and Comp. Eng.
Univ. of Minnesota Chicago, IL 60607, USA

Abstract
Constraint satisfaction during partitioning and placement of VLSI circuits is an important problem, and effective techniques to address it lead to high-quality physical design solutions. This problem has, however, been cursorily treated in previous partitioning and placement research. Our work presented here addresses the balance-ratio constraint, and is a crucial first step to an effective solution to the general constraint-satisfaction problem. In current iterative-improvement mincut partitioners, the balance-ratio constraint is tackled by disallowing moves that violate it. These methods can lead to sub-optimal solutions since the process is biased against the movement of large cells and clusters of cells. We present techniques for an informed relaxation process that attempts to estimate whether relaxing the constraint temporarily will ultimately benefit the mincut objective. If so, then a violating move is allowed, otherwise it is disallowed. The violations are corrected in future moves so that the final solution satisfies the given constraint. On a set of ACM/SIGDA PROUD benchmark circuits with actual cell sizes, we obtained up to 38% and an average of 14.5% better cutsizes with as little as 13% time overhead using our techniques compared to the standard method of not allowing any relaxation.

1. Introduction
Motivation
The partitioning problem for any given objective is inherently a constraint-driven one. The most common constraint is the balance-ratio of total sizes that the two subsets of a partition must satisfy. Other frequently encountered partitioning constraints include limits on the sizes, pins and prices in multiple-FPGA partitioning [7, 8], balance ratios for multiple cell types (this occurs in FPGA placement and in pin assignment), upper bound on the deterioration of an objective, say, wire-area minimization, of an initial placement when the circuit is hierarchically re-partitioned for another objective, say, timing minimization (this arises in multi-objective placement), and even distributions of external nets and internal pins among the subcircuits of a partition (to avoid congestion [10]). Constraint satisfaction during partitioning and placement of VLSI circuits is an important problem, and effective techniques to address it lead to high-quality physical design solutions. This problem has, however, been cursorily treated in previous partitioning and placement research. For example, a widely used practice for constraint-satisfaction is to disallow cell moves that violate a constraint, which over-constrains the partitioner leading to sub-optimal results. Our work presented here addresses the balance-ratio constraint, and is a crucial first step to an effective solution to the general constraint-satisfaction problem. During the hierarchical partitioning of a circuit in partition-driven placement tools [11], the pre-specified balance-ratios are very important constraints that lead to an area-minimal balanced layout of the circuit (see Fig. 1).

This balance-constrained partitioning problem can be stated as follows. Given a netlist G which describes cell connectivities and cell sizes in a circuit, construct k sub-circuits of G with a balance-ratio of r1 : r2 : ... : rk, and an acceptable tolerance of ±ε such that some objective functions are optimized. In an attempt to satisfy the balance-ratio, however, movement of larger cells and clusters can be restricted thus leading to sub-optimal results. Figures 2 illustrates these problems in which restricted movements of large cells and clusters lead to larger cutsizes.

PastWork
All the general iterative-improvement partitioning techniques such as FM [3], LA [4], PROP [1], GFM [5], GMetis [6], and hMETIS [12] do not allow any type of relaxation. They assume unit-size cells, which do not work well for general/macro-cell circuits. Wei and Cheng proposed a ratio-cut method and applied it to general-cell circuits with certain pre-specified balance-ratios [9]. Their results, however, became much worse as the balance-ratio went closer to 1:1. Until recently, partitioning techniques with minor relaxations were employed in multiple-FPGA partitioning which generally has many constraints [7] [8]. Kuznar, Brejle and Kozminski [7] used the cost function to levy large penalties on violating moves. This causes the partitioner to accept very few violating moves, which most likely produces results identical to not allowing any relaxation—there are no results in [7] comparing relaxed and non-relaxed formulations. Woo and Kim [8] relaxed constraints by reducing the originally specified constraints (e.g., doubling the numbers of pins required in each FPGA) for some pre-specified passes. The partition process may terminate with violations if the number of pre-specified passes is greater than or equal to the total number of passes, in which case the process has to start over with a different initial partition. These two relaxed partitioning techniques provided ad hoc solutions to tackling constraints, that were specific to the particular problem. In contrast, our approach provides a general framework for partitioning in the presence of constraints and uses informed intermediate relaxation techniques for this purpose.

New Approaches
We propose branch-and-bound approaches to constraint-driven partitioning that allows intermediate constraint relaxation during cell moves in order to ensure better results. The new methods can be generally stated as follows. In the first approach, on every cell move which results in constraint violation, the cost and benefit of allowing the violating move is estimated. The accepted moves are those which potentially give better results (compared to disallowing the moves) taking into consideration that a valid cut-point cannot be established until the violation is corrected in future “reverse”
moves. In the second, lower time-complexity technique based on statistical studies, a “benefit factor” and “acceptance threshold” are computed for a violating move. Depending on the stage of the partitioning process and the location and status of the violating cell and its neighbors, the move is accepted if the benefit factor is greater than the acceptance threshold. In order to keep the computation complexity reasonable, we do not backtrack in the move process using either of the above approaches; hence it becomes very critical for our benefit estimator to be accurate.

The rest of the paper is organized as follows. In Sec. 2, we formally state the balance-constraint based partitioning problem, and recap the relevant gain formulation of PROP [1]. Section 3 shows examples of the advantage of intermediate relaxations of the balance constraint, and then discusses general approaches to relaxed constraint-satisfaction. Section 4. presents the particular techniques for tackling the balance-ratio constraint in the context of the PROP partitioning method. In Sec. 5, we present experimental results, and conclude in Sec. 6.

2. The Bi-partitioning Problem with Balance Constraints

The balance-constrained 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 vertices that represent cells or modules in the circuit, and $E = \{e_1, e_2, ..., e_m\}$ is the set of hyperedges which represent nets in the circuit; hence, $n$ and $m$ 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. The area of each cell is denoted as $a(v_i)$, where $1 \leq i \leq n$. 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 crossing $V_1$ and $V_2$ by $E_{\text{cut}}$, and the total size of $V_1$, $V_2$ by $|V_1|$, $|V_2|$ respectively. Therefore, $|V_j| = \sum_{i \in V_j} a(v_i)$ for $j = 1$ and 2.

There are one or more objectives and constraints that any partitioner must achieve and comply with, respectively. One of the most common objectives is to minimize the cutsize of the cutset which is defined as $\text{cut size} = \sum_{i \in E_{\text{cut}}} w(n_i)$, where $w(n_i)$ is the weight or cost of net associated with $n_i$. Based on this objective, every cell $u$ in the circuit is assigned a gain $g(u)$. This gain is defined differently for different partitioners like FM, LA, and PROP, and have different degrees of sophistication in terms of the given objective(s) by allowing intermediate constraint relaxation.

3. Tackling Constraints with Intermediate Relaxation

3.1. The Need for Relaxation

Most partitioning methods that do not relax size-balance constraints work reasonably for circuits such as gate arrays that have uniform cell sizes. However, when they are applied to circuits with general-cells or macro-cells that have varying sizes, these algorithms suffer from a propensity to freeze the movement of large cells that would immediately violate the balance constraint. To illustrate this problem, consider the example shown in Fig. 2. Let $A/8$ be the total area of the circuit. Cells labeled from 1 to 4 have areas of $A/8$ each, and cells labeled from 5 to 12 have areas of $A/16$ each. The balance ratio $r$ is 0.5 (50%-50%) with a tolerance of 0.1 (10%). Without any relaxation, cells 1 to 4 will be “stuck” in their initial subsets due to their large sizes, while cells 5 to 12 have more freedom to move around as shown in Fig. 2(b). However, if we allow a relaxation of at least $A/8$ (relax $= 1/8$), cells 1 to 4 can move freely as dictated by their gains, and this results in the optimal solution shown in Fig. 2(c) that satisfies the $r = 0.5$ constraint.

Freeing up the movement of large cells is not the only advantage of a “relaxed” partitioner. Relaxed partitioning also helps in removing natural clusters that straddle the cutline by blocks of unidirectional moves that temporary violate the balance constraint; in a non-relaxed partitioner, these clusters can get locked in the cutset due to alternating moves in the two directions.

The idea behind our relaxation approach is illustrated in Fig. 3. Figure 3(a) shows a portion of a plot of the cutsize versus cell moves that are all non-violating; therefore, each point along the curve is a valid prefix point, with $M$ being the mincut point. The best moves following $M$ are violating moves that would actually continue to reduce the cutsize, but are rejected in a non-relaxed
3.2. General Formulation

The new approach to constraint relaxation for any given objective and any given constraint can be stated in the following general framework. Let $T$ be the current set of the most recent violating moves (from $V_i$ to $V_T$, where $V_T = V - V_i$). Let $R$ be a corresponding set of required reverse moves from $V_T$ to $V_i$ of the best cells in $V_T$ that can correct the violation due to $T$. Thus $R$ represents moves that can correct the violations due to $T$, and in the process will cause the minimum deterioration of the optimization objective (see Fig. 3). Initially, before any violating moves have been made, $T = R = \emptyset$. Let $v$ be the best move to make at the current point in the move process. If $v$ does not add to the accumulated violation due to $T$, then we move $v$. Otherwise, we search for a set of cells $\text{rev}(v), \text{rev}(v) \cap R = \emptyset$, that can be reverse-moved in order to compensate for $v$’s violation such that

$$\text{adv}(v) + \text{adv}(R \cup \text{rev}(v)) | v > \text{adv}(R)$$

(2)

where $\text{adv}(y)$ (read “advantage of $y$”) is the estimated improvement in the optimization objective due to the movement of cell $y$ (adv($y$) can be negative), and for a set $S$ of cells, $\text{adv}(S) = \sum_{y \in S} \text{adv}(y)$. Further, $\text{adv}(y | x)$ is the estimated improvement obtained by moving cell $y$ after cell $x$ has been moved—adv($y$) can be positively or negatively affected by cell $x$’s move, and this is generally negative if $x$ and $y$ on opposite sides of the partition. Further, $\text{adv}(S | x) = \sum_{y \in S} \text{adv}(y | x)$. The LHS (left-hand side) of Ineq. 2 is the estimated objective improvement on pursuing the path of moving $v$ followed by $R \cup \text{rev}(v)$ to correct the total violation, while the RHS is the estimated objective improvement obtained by only moving cells in $R$ to correct the current violation; note that a legal partition point cannot be taken when there are no violations. Cell $v$ would then be moved if $\text{rev}(v)$ exists, otherwise the best cell from $R$ is moved.

3.3. Application to Balance Constraints

Our current approaches apply to constraints in the size ratio between $V_i$ and $V_T$, and the objective is the minimum cutsize between the two subsets. They use the fundamental idea given by Ineq. 2.

Let $(u_1, u_2, \ldots, u_m)$ be the ordering of the unlocked cells in $V_T$ by decreasing gains. If $T$ is the most recent set of violating moves from $V_i$ to $V_T$ and $z = \sum_{v \in T} a(v)$ is the total size of the violations, then $R = \bigcup_{j=1}^{m} \{u_j\}$ where $|\bigcup_{j=1}^{m} u_j| \geq z$, and $|\bigcup_{j=1}^{m} u_j| < z$. $\text{adv}(y) = g(y)$ is the probabilistic gain of cell $y$. If $v$ is the current best node of size $a(v)$ whose move adds to the violation, then $\text{rev}(v) = \bigcup_{j=1}^{m} u_j$, where $|\bigcup_{j=1}^{m} u_j| \geq a(v)$ and $|\bigcup_{j=1}^{m} u_j| < a(v)$. The move of $v$ is accepted if (following the general condition of Ineq. 2)

$$(1 - \alpha)g(\text{rev}(v)) + g(v) - \alpha g(R) > 0$$

(3)

The negative term in Ineq 3 corresponds to the average-case decrease in gains of cells $y \in R$ due to an average of $\alpha$ connections between $v$ and $y$.

Further, in order to obtain a fair number of points at which legal mincuts can be considered, reverse moves (with minimum deterioration of the mincut) will have to periodically occur to correct the violations. Hence, violations are allowed to occur until a relaxation limit $\text{relax}$ is reached, and reverse moves are made after this point.

4. Incorporating Relaxations in PROP

4.1. Estimation Method

We present here an accurate version of the estimator in the LHS of Ineq. 2. There are two sources of this accuracy: (i) a look-ahead technique that incorporates the decrease in the gain of future reverse-moves $R'$ (moves beyond $R \cup \text{rev}(v)$) due to the current violating move $v$, and (ii) an exact calculation of the decrease in gain in $R \cup \text{rev}(v) \cup R'$ due to $u$’s move.

Due to the presence of clusters straddling the cutline, a violating move from side $V_i$ will most likely be followed by further violating moves from $V_i$. In order to accurately estimate the effect of moving $v$ on all reverse moves that will potentially need to be made after the current block of violating moves end, we associate a set of potential future reverse-move cells $C$ in $V - (R \cup \text{rev}(v)) = \emptyset$ corresponding to future violating moves. In this manner, the estimator can “look-ahead” further, and hence predict more accurately the expected improvement in the cutsize after the current block of violating moves and the reverse correcting moves are completed. Specifically, $R'$ is the highest-order subset of $V_T - (R \cup \text{rev}(v))$ ordered by decreasing gains. The size of $R'$ depends on the relaxation limit (relax), and incrementally changes as the violating moves occur, $|R \cup \text{rev}(v) \cup R' \leq \beta \text{relax}|$ where $0 \leq \beta \leq 1$, is the lookahead parameter, and $\text{relax} = \text{relax}|V_i|$ is the size of the total relaxation allowed.

We next describe a precise calculation of the decrease in the gains of cells in $R \cup \text{rev}(v) \cup R'$ due to the move of cell $v$. Recall
again that since \( v \) is a violating move, its effect on the gains of all future reverse-move cells need to be considered, since a valid cut point can only be taken for the maximum prefix computation after all these reverse-moves have been made. If \( \text{rev}(v) \) corresponding to the current violating move of \( v \in V_2 \) can be found in \( V_2 \), then \( g(\text{rev}(v)) = \sum_{y \in \text{rev}(v)} g(y) \), where \( g(y) \) is the probabilistic gain of cell \( y \). Assuming \( y \in V_2 \), each component \( g_n(y) \) of \( g(y) \) can be reformulated as
\[
g_n(y) = g_n^+(y) - g_n^-(y)
\]
where from Eq. 1, \( g_n^+(y) = \left( \prod_{y' \in E, x \in y} p(y, x) \right) / p(y) \) and \( g_n^-(y) = \left( \prod_{y' \in E, x \notin y} p(y', x) \right) / p(y) \).

The “conditional” gain \( g(y | v) \) of a cell \( y \) in \( R \cup \text{rev}(v) \cup R' \) can be accurately computed as
\[
g(y | v) = g(y) - \sum_{n_1 \in R \cup \text{rev}(v) \cup R'} g_n^-(y, v)
\]
The rationale behind this computation is that, if there is a currently gain component \( g_n^+(y) \) of some \( y \in R \cup \text{rev}(v) \cup R' \) on a common net \( n_1 \), between \( y \) and \( v \), then \( g_n^+(y) \) will become 0 when \( v \) is moved to \( V_2 \) and locked. Thus the estimating condition becomes
\[
g(v) + g(\text{rev}(v)) - \sum_{y \in R \cup \text{rev}(v) \cup R'} \sum_{n_1 \in R \cup \text{rev}(v) \cup R'} g_n^-(y) > 0
\]
On every violating move attempt, the benefit is estimated based on Ineq. 6. A violation is rejected and reverse moves started if either Ineq. 6 is not satisfied, or \( z + a(v) > \text{relax} \) where \( z \) is the total size of most recent violating moves (\( T \)), and \( a(v) \) is the size of the current violating cell \( v \). No violations are permitted during the reverse process in order to guarantee the availability of a fair number of legal positions at which the maximum prefix can be computed. The relaxation resumes once violations are corrected. If the estimation is accurate enough, the cost of the cutsize after moving the current block of violating moves followed by the required reverse-moves, will be lower than disabling any or fewer violating moves.

A high-level description of the estimation algorithm on a violating move is given in Fig. 4. The timing complexity of GainEstimate() is \( O(|\text{rev}(v)| + \sum_{n \in V_2} n_1) \). On the average, \( |\text{rev}(v)| \) will be a constant, and a constant fraction of the \( n \) moves attempted are violations, the additional complexity of PROP-REX 알고리즘 (PROP with relaxation using GainEstimate) will be at most \( \frac{1}{2} \sum_{n_1 \in E} \rho_{n_1} = O(n \rho_{\max \rho} \rho_{\max}) \) where \( \rho_{\max} \) is the number of pins on net \( n \), \( \rho_{\max} \) is the maximum number of pins on any cell in \( G \), and \( \rho_{\max} \) is the maximum number of pins on any net in \( G \).

### 4.2. Weighted Benefit Method

In an attempt to reduce the timing complexity of PROP-REX 알고리즘, we investigated a weighted benefit method that captures heuristic information on the benefit of making a violating move. We denote \( B(v) \), \( 0 \leq B(v) \leq 1 \), as the total weighted “benefit” of moving a violating cell \( v \) with size \( a(v) \) from \( V_2 \) to \( V_1 \). \( B(v) \) is defined as \( B(v) = \sum_{i=1}^{c} w_i b_i \), where \( c \) is the total number of factors considered, \( w_i, 0 \leq w_i \leq 1 \), is the weight on benefit \( b_i, 0 \leq b_i \leq 1 \), and \( \sum_{i=1}^{c} w_i = 1 \). We considered three benefit factors. The first factor is the normalized gain difference between the highest-gain cells \( v \in V_2 \) and \( u \in V_1 \). The second factor is the ratio of the total “available” violation (which is \( \text{relax} - z - a(v) \), where \( z \) is the total violation so far and \( a(v) \) is the size of the violating cell \( v \)) and the maximum allowable violation \( \text{relax} \). The third factor is the ratio of the total size of free cells in \( V_1 \) that will result after correcting all violations and \( |F_1| \), the current total free cell size in \( V_1 \). Hence \( B(v) \) is given by
\[
B(v) = w_1 (\phi(g(v) - g(u))) + w_2 (1 - \frac{z + a(v)}{\text{relax}}) + w_3 (1 - \frac{z + a(v)}{|F_1|})
\]
where \( \phi(\cdot) \) is a linear function that normalizes the gain difference to a number between 0 and 1. Based on our experiments, the best values for each weight has been determined to be \( w_1 = 0.5 \), \( w_2 = 0.25 \) and \( w_3 = 0.25 \). Essentially \( B(v) \) captures the notion that a violating cell should have a high probability of being allowed to move, if the gain difference between \( v \) and \( u \) is high, if the current violations are far from the relaxation limit, and the move process

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Cells</th>
<th>Nets</th>
<th>IO</th>
<th>Avg</th>
<th>Stdv</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_large</td>
<td>21578</td>
<td>23384</td>
<td>64</td>
<td>1845.4</td>
<td>1069.2</td>
<td>4176</td>
<td>16</td>
</tr>
<tr>
<td>avg_small</td>
<td>20978</td>
<td>22124</td>
<td>64</td>
<td>2000.1</td>
<td>1150.2</td>
<td>4176</td>
<td>16</td>
</tr>
<tr>
<td>busmed</td>
<td>63514</td>
<td>5742</td>
<td>97</td>
<td>1000.1</td>
<td>1102.0</td>
<td>4176</td>
<td>16</td>
</tr>
<tr>
<td>primary1</td>
<td>859</td>
<td>10</td>
<td>107</td>
<td>14004</td>
<td>1175</td>
<td>30000</td>
<td>25</td>
</tr>
<tr>
<td>industry1</td>
<td>12637</td>
<td>13419</td>
<td>495</td>
<td>32935</td>
<td>18220</td>
<td>116480</td>
<td>1</td>
</tr>
<tr>
<td>industry3</td>
<td>15406</td>
<td>21924</td>
<td>374</td>
<td>93494</td>
<td>36463</td>
<td>237568</td>
<td>64</td>
</tr>
<tr>
<td>s13207*</td>
<td>8772</td>
<td>6651</td>
<td>152</td>
<td>8847</td>
<td>1154.0</td>
<td>900</td>
<td>16</td>
</tr>
<tr>
<td>s15580*</td>
<td>10470</td>
<td>10383</td>
<td>101</td>
<td>891.5</td>
<td>86.4</td>
<td>900</td>
<td>16</td>
</tr>
<tr>
<td>s35932*</td>
<td>18148</td>
<td>17828</td>
<td>335</td>
<td>882.7</td>
<td>122.4</td>
<td>900</td>
<td>16</td>
</tr>
<tr>
<td>s38417*</td>
<td>23949</td>
<td>23843</td>
<td>134</td>
<td>8951</td>
<td>65.9</td>
<td>900</td>
<td>16</td>
</tr>
<tr>
<td>s38584*</td>
<td>20995</td>
<td>20717</td>
<td>290</td>
<td>887.8</td>
<td>103.2</td>
<td>900</td>
<td>16</td>
</tr>
<tr>
<td>v9234*</td>
<td>5866</td>
<td>5844</td>
<td>41</td>
<td>893.8</td>
<td>73.7</td>
<td>900</td>
<td>16</td>
</tr>
<tr>
<td>sico</td>
<td>664</td>
<td>408</td>
<td>62</td>
<td>2862.3</td>
<td>1125.0</td>
<td>3551</td>
<td>1</td>
</tr>
<tr>
<td>struct</td>
<td>1852</td>
<td>1920</td>
<td>64</td>
<td>14607.0</td>
<td>518.8</td>
<td>2320</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 1: Benchmark circuit characteristics. Circuits with a * suffix are gate-array circuits for which there are only two distinct sizes (gates and pads). **Cells** is the total number of cells (including I/O pads). **Nets** is the total number of nets, **IO** is the number of I/O pads, **Avg** is the average cell size, **Stdv** is standard deviation of cell sizes, and **Max/Min** are the maximum/minimum cell sizes in the circuit.
is far from the point at which all cells are locked.

A moving violating move with benefit $B(v)$ will be accepted if it is above a certain threshold $T(v)$ that depends on the difference $\Delta \text{cut}$ between the current cutsize $c_{\text{cut}}$ and the minimum cutsize $c_{\text{min}}$ found so far. This is best modeled by a $\text{tanh}$ function given by

$$T(v) = f(\Delta \text{cut}) = 1 - \frac{1}{2}(\text{tanh}(\frac{\Delta \text{cut}}{f_h} - f_v) + 1) \quad (8)$$

where $f_h$ and $f_v$ are control variables which reshape the function curve as shown in Fig. 5(a).

The weighted benefit method is applied selectively along certain portions of the cutsize curve. The main aim is to reduce the time overhead of using GainEstim at every moving violation, without sacrificing the cutsize significantly. The critical junctures to perform accurate benefit estimations for violating moves are when the current cutsize is decreasing, and is either near the minimum cutsize found so far, or it has just decreased below $c_{\text{min}}$ (see Fig. 5(b)). These portions are critical since new mincut points will be occurring in these regions, and it is desirable to have valid prefix points here. Thus if the possibility of violating moves present themselves in these portions of the cutsize curve, then it is important to evaluate their benefits accurately by applying GainEstim in these sections.

In the regions where $c_{\text{cut}}$ is increasing beyond $c_{\text{min}}$, or is decreasing but still far away from $c_{\text{min}}$, it is not necessary to have valid prefix points. Thus the less accurate, though still informed, weighted benefit method can be used. At the same time, it is beneficial to allow more violations in portions where $\Delta \text{cut}$ is increasing, or is decreasing but large, in an effort to decrease $c_{\text{cut}}$ below $c_{\text{min}}$. Also, so that valid prefix points can be found soon after $c_{\text{cut}}$ decreases below $c_{\text{min}}$, it is essential to make fewer violating moves when $\Delta \text{cut}$ is small and $c_{\text{cut}}$ is decreasing toward $c_{\text{min}}$. All these desirable characteristics are facilitated by the $\text{tanh}(\cdot)$ based threshold function as shown in Fig. 5(a).

The timing complexity per violating move of the weighted benefit method is a constant. Thus the timing complexity of PROP (PROP with relaxation, using both GainEstim and weighted benefit methods) is reduced to $O(n c_{\text{cut}} + n e_{\text{net}})$ where $\gamma$ is the fraction of violating moves occurring at $c_{\text{cut}} \leq c_{\text{min}}$. Empirical results show $\gamma$ to be around 1/3.

5. Experimental Results

The relaxation algorithms were incorporated into PROP. Experiments have been done on ACM/SIGDA benchmark circuits which were translated from the PROUD format in which actual cell sizes are given; characteristics are given in Table 1.

All experimental results presented here were run with a tolerance ($t$) of 0.005 (0.5%) for a balance ratio ($r$) of 0.5 (50%-50%), 0.45, 0.4 and varying relaxation limits ($\text{relax} = 0.01$ to 0.05), on a SPARCStation 20. The first set of results given in Table 2 compare FM (20 and 60 runs—60 runs make its speed comparable to PROP with 20 runs), PROP (both without relaxations), and PROP with a “blind” relaxation, where all violating moves are just “blindly” accepted and reversed once the relaxation limit is reached (0.05 for the results shown), both with 20 runs. PROP without relaxation is better than FM-20 by about 35% in mincut and 42.5% in average-cut, and is better than FM-60 by about 25%.
Table 3: Cutsizes and improvements on benchmark circuits achieved by PROP-REX$_{relax}$ (left half) and PROP-REX$_{no-relax}$ (right half) for $r = 0.5$ and $t = 0.005$.

in mincut and 42% in average cut. This establishes that PROP is indeed a very good partitioning algorithm for circuits with varying cell sizes ([1] established this fact for circuits with unit cell sizes), and is thus a good candidate for trying to improve further using our new relaxation techniques. PROP with blind relaxation is only 2% better in mincut than PROP without relaxation, but is slightly worse in the average cut. This underscores the need to have good benefit estimation techniques for violating moves—only allowing relaxation in the move process is not enough. Table 3 shows that the new benefit estimation methods perform well in predicting and allowing only “good” violating moves.

Table 3 compares the results of PROP without relaxation to PROP-REX$_{relax}$ and PROP-REX$_{no-relax}$, for a look-ahead factor $(3)$ of 0.5 for 20 random runs. From Table 3, PROP-REX$_{relax}$ achieves significant improvements on large general-cell circuits such as avg$_{large}$, avg$_{small}$, industry2, and industry3a by as much as 37%. Overall, there are improvements of 14.55% and 16.25% in mincut over all circuits for relax = 0.03 and relax = 0.05, respectively. This indicates that the estimator does work well in predicting “good” violating moves. PROP-REX$_{relax}$ is about 1.43 times slower than PROP (for relax = 0.03).

From Table 3, PROP-REX$_{no-relax}$ achieves significant improvements for general-cell circuits; for circuit avg$_{large}$, there is an improvement of 38% in mincut for relax = 0.05. Over all circuits, PROP-REX$_{no-relax}$ achieves an improvement of 14.5% in mincut for relax = 0.03. These improvements are comparable to those of PROP-REX$_{relax}$, though PROP-REX$_{relax}$ is only about 1.13 times slower than PROP (for relax = 0.03). We have also obtained experimental results for $r = 0.45$ and $r = 0.4$, that show that PROP-REX$_{no-relax}$ achieves an overall improvement of 10% for both balance ratios.

6. Conclusions

We proposed a new general non-backtracking branch-and-bound approach to the important problem of constraint-driven partitioning, and then presented specific techniques for tackling one of the most important and commonly applied constraints, a pre-specified balance-ratio on the sizes of the subcircuits of a partition. Effective constraint satisfaction was either overlooked in past partitioning work (e.g., [1], [3], [4], [5] and [6]), or was treated cursorily in others ([7], [8]). Our new techniques estimate the “goodness” of accepting violations compared to disallowing them, based on probabilistic cell gains, look-aheads for future violations, and weighted benefits. Incorporating the new methods within the state-of-the-art partitioner PROP yields improvements of up to 38% on general-cell circuits, and 18% on gate-array circuits with an average improvement of 14.5%, and a timing overhead of only around 13%.

The work presented in this paper represents a crucial first step towards formulating effective relaxation techniques for a variety of constraints that occur naturally in the physical design of VLSI circuits. In future work, relaxation methods will be developed for other important constraints, such as those mentioned in Sec. 1., and a precise general formulation will be developed that can be effectively applied to most constraints.

References