On Wirelength Prediction Using the Net-cut Objective [1]

Maogang Wang and *Majid Sarrafzadeh*
Department of Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208
Tel: (847) 491-7378
Fax: (847) 467-4144
Email: mgwang,majid@ece.nwu.edu

Abstract

The multi-level hierarchical technique is regarded indispensable for solving today's complex VLSI placement problem without sacrificing quality. How to solve a hierarchical placement problem becomes very important. Net-cut and wirelength are widely used in hierarchical placement problems. In this paper we study the behavior of these two objectives in the hierarchical placement problem.

We defined α's to express the difference between wirelength and net-cut at different hierarchical levels. We proved that the net-cut objective is a good approximation of length at coarser hierarchical levels. At finer levels the net-cut objective gets further from the wirelength objective. Experimental results are shown to support this claim. Thus a good way to minimize wirelength for a top-down approach is to consider net-cut at early hierarchical levels and switch to wirelength later. We proposed a "+1 *level clustering*" technique. Experiments show that this technique can effectively combine the advantage of minimizing net-cut (fast) and wirelength (accurate) together in later hierarchical levels. Finally we showed that the percentage of external nets is important to determine where we should switch from the net-cut objective to the wirelength objective. Experimental data showed that if more than $20\% - 30\%$ nets are external, wirelength should be considered in the optimization objective; Otherwise, net-cut is a reasonable estimate of wirelength.

[1]This work was support in part by NSF grant MIP-9527389.

1 Introduction

Placement is a classical problem in VLSI physical design. A lot of effective placement algorithms have been proposed in the last twenty years [10, 15, 14, 7, 12]. As the VLSI circuit size becomes larger and the technology goes into the range of deep sub-micron, it becomes hard to solve the placement problem flat. The multi-level hierarchical technique is regarded indispensable for solving today's complex VLSI placement problem without sacrificing quality. In [12], Sarrafzadeh and Wang showed that solving a hierarchical placement problem helps to reduce the size of the solution space of the original placement problem. At a given hierarchical level, we partition the whole chip area into several global bins. Then the hierarchical placement problem is to place cells at the center of each global bin to optimize a certain cost function. Most state-of-the-art placement tools [10, 12, 14, 7] employ a hierarchical approach, this include top-down annealing approaches [14, 13, 12] and recursive quadratic methods [10, 7]. They solve the placement problem successively at different hierarchical levels. Two objectives, the net-cut objective and the wirelength objective are commonly used when solving the hierarchical placement problem. Optimizing the net-cut objective aims to reduce the number of connections between global bins while optimizing the wirelength objective aims to reduce the global interconnection length which results in improved routability. The wirelength objective is closer to the original placement problem. Indeed, wirelength metric is globally consistent with routability (congestion) improvement [16]. However, since the net-cut objective is correlated with the wirelength objective and is better studied, the net-cut objective is also widely used in the hierarchical placement problem.

There are different ways to solve the hierarchical placement problem at each level of hierarchy. In [12], the authors use a simple simulated annealing technique with the wirelength objective. TimberWolf [14, 13] uses a clustering technique with the net-cut objective to first form cell clusters. Then it uses simulated annealing to place these clusters in global bins using wirelength objective. Gordian and other top-down quadratic placement algorithms first use analytical methods to get approximate locations of cells with minimized wirelength. Then cells are placed in global bins based on their approximate locations and possibly the net-cut information. Although both the wirelength and the net-cut objective are used in different hierarchical placement algorithms, the relationship between these two objectives are still not well understood. Questions such as how good is the net-cut objective and how well it estimates wirelength remain unanswered. In this paper, we analyze the relationship between the net-cut and the wirelength objective in hierarchical placement and show scenarios where net-cut is a good prediction of wirelength and where it is not.

In this paper we first proved that the net-cut objective is a good approximation of length at coarser hierarchical levels. At finer levels the net-cut objective gets further from the wirelength objective. Thus we should focus on wirelength at finer levels. Extensive experimental results are shown to support our analysis. We also show how to use existing partitioning tools (e.g., hMetis [8, 9]) to minimize wirelength. An algorithm to combine wirelength and net-cut is proposed. Finally we show how to determine the hierarchical level where we should switch from the net-cut objective to the wirelength objective.

The rest of the paper is organized as follows: In Section 2, we formulate the problem and formally state the questions. In Section 3, we theoretically analyze the relationship between the

net-cut and the wirelength objective and show results from experiments to support our analysis. In Section 4, we will explore methods of combining the net-cut and the wirelength objective together to effectively solve the hierarchical placement problem, followed by the conclusion in Section 5.

2 Problem Formulation

As circuits get larger, the placement problem for VLSI physical design can only be solved effectively using hierarchical approaches. The size of the solution space grows exponentially with the size of the circuit. Thus solving the hierarchical placement problem is much faster than solving the original problem. VLSI designers also tend to design circuits hierarchically. This also gives hierarchical placement algorithms big advantages over flat placement algorithms. In this section, we will formulate the hierarchical placement problem.

A typical top-down placement approach is based on recursive circuit partitioning. It repeatedly divides a given circuit into subcircuits to optimize a given partitioning objective. At each level, the given layout area is partitioned in either the horizontal or the vertical direction or both. Each subcircuit is assigned to a partition. Recursive partitioning is repeated until each subcircuit contains a small number of cells.

An arbitrary hierarchical placement algorithm may not be based on a partitioning technique. Thus we use the concept of global bins instead of the concept of partitions to allow more general analysis and discussion. At a given hierarchical level, we divide the layout area into N_b rectilinear regions, each of these regions is called a global bin. Assume we have r rows and c columns of global bins ($N_b = r \times c$). We label the global bin at i_{th} row and j_{th} column as B_{ij}. From the top left global bin, the labels are $B_{11}, B_{12}, B_{13}, ..., B_{ij}, ..., B_{rc}$. The center of global bin B_{ij} is denoted by $C_{B_{ij}} = (x_{B_{i,j}}, y_{B_{i,j}})$. Figure 1 shows an example where we have $4 \times 4 = 16$ global bins.

In this paper we assume that we are given a circuit denoted $Ckt(\mathcal{C}, \mathcal{N})$, which consists of a set of cells $\mathcal{C} = \{C_i | i = 1, \ldots, |\mathcal{C}|\}$, and a set of nets $\mathcal{N} = \{N_i | i = 1, \ldots, |\mathcal{N}|\}$. Each *net* N_k consists of a set of terminals. Terminals are given a location on the surface of a layout area by the placement process. The location of a terminal is represented by $s_{i,k} = (x_{i,k}, y_{i,k})$, thus $S_k \subset \Re^2$, where $\Re$ is the set of real numbers. The rectilinear distance between two terminals $s_{i,k}, s_{j,m}$ is $\|s_{i,k}, s_{j,m}\| = |x_{i,k} - x_{j,m}| + |y_{i,k} - y_{j,m}|$. Similarly, each cell C_j contains a set of terminals $S(C_j) = \{s_{j,k} \mid i = 0 \ldots |S(C_j)|\}$. Let the location, on the plane, of the cell C_j be denoted by (x_{C_j}, y_{C_j}). Then the location of a terminal $s_{j,k} \in S(C_j)$ is the same as the location of the parent cell (x_{C_j}, y_{C_j}) in the hierarchical placement problem. The wirelength of a net N_i is defined as the half perimeter of the bounding-box for net N_i. The total wirelength of the placed circuit is the summation of the wirelength for all the nets in the circuit.

In the hierarchical placement problem, each global bin $B_{i,j}$ contains a set of cells $P_{i,j} \subset \mathcal{C}$. For any cell in a global bin, the location of the cell is set to the center of that bin. $\forall C_k \in P_{i,j}, (x_{C_j}, y_{C_j}) = (x_{B_{i,j}}, y_{B_{i,j}})$. Cells are placed into global bins to minimize the total wirelength. In order to prevent all the cells from being placed in the same global bin (zero total wirelength), the balancing constraint has to be imposed. The balancing constraint for a certain hierarchical level which has N_b global bins can be described as: $(1-u)\frac{|\mathcal{C}|}{N_b} < |P_{i,j}| < (1+u)\frac{|\mathcal{C}|}{N_b}$ where

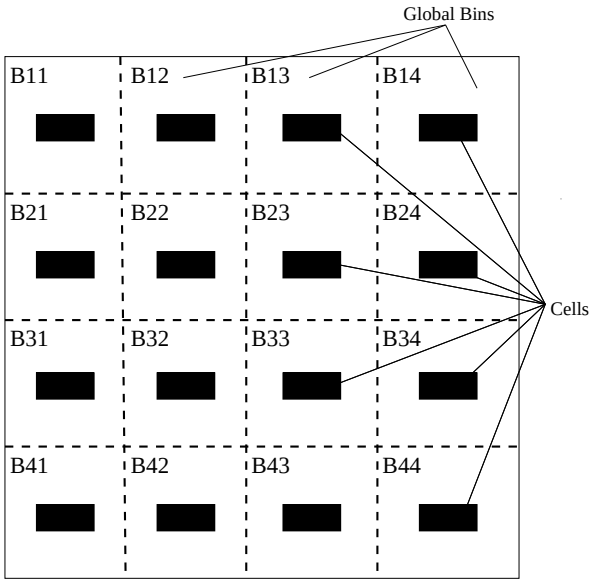


Figure 1: Hierarchical placement and global bins.

$0 < u < 1$ is the unbalancing factor.

Net N_i is not cut if and only if all the terminals of N_i are located in the same global bin. The net-cut at a given hierarchical level is defined as the total number of cut nets. This definition is consistent with net-cut definition for a N_b-way partitioning problem. We call all the uncut nets the *internal nets* since they are located inside one global bin. We call all the cut nets the *external nets* since they span more than one global bin.

Generally, a top-down hierarchical placement approach will start from the first hierarchical level h_1 and go down to lower hierarchical levels. The detailed procedure is described as the following: It solves the hierarchical placement problem at the current hierarchical level h_i which has N_{bi} global bins. Then it will go to the next hierarchical level h_{i+1} by splitting each global bin in level h_i into g smaller global bins. Thus the level h_{i+1} will have $N_{b(i+1)} = gN_{bi}$ global bins in total. It keeps doing this until the number of cells in each global bin is less than a certain value. Figure 2 illustrate the basic flow of such an approach.

3 Relationship Between the Net-cut and the Wirelength Objective

The total wirelength is the objective to minimize in the hierarchical placement problem, for it closely relates to routability [16]. However, the net-cut objective is better researched and easier to optimize. There are a number of partitioning tools which can minimize the net-cut objective very effectively and efficiently [8, 5, 2, 3, 6, 9]. Intuitively, minimizing net-cut in the hierarchical placement reduces the connections between global bins. Thus it tends to reduce the wirelength as well. Since the pure net-cut objective does not consider the geometrical information, minimizing the wirelength objective produces much better wirelength results than

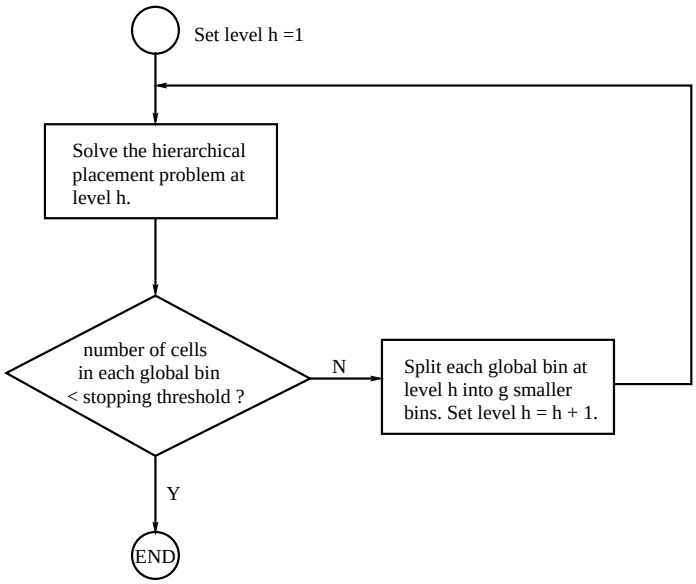


Figure 2: Working flow of a top-down hierarchical placement approach.

minimizing the net-cut objective.

As an approach, we can first use the net-cut objective to partition the circuit into N_b subcircuits. Then we place each subcircuit in one global bin to minimize the wirelength. This is the minimal wirelength hierarchical placement with the optimized net-cut objective. We call this hierarchical placement a *net-cut optimized placement* and the hierarchical placement obtained by minimizing wirelength a *wirelength optimized placement*.

In this section, we will first theoretically analyze the relationship between the wirelength and the net-cut objective at different hierarchical levels and compare the net-cut optimized placement with the wirelength optimized placement. Related experimental results will be presented after the theoretical analysis.

The net-cut is the number of nets cut by the global bins. In order to make it comparable to the wirelength objective, we normalize the wirelength cost using the dimension of the global bins. Since all the terminals are located at the centers of global bins, the horizontal distance between two terminals is multiples of the global bin width and the vertical distance between two terminals is multiples of the global bin height. We define the normalized horizontal distance between two terminals to be the number of global bins between these two terminals in the horizontal direction. The normalized vertical distance can be similarly defined.

When the given hierarchical level has only two global bins, the net-cut and the normalized wirelength objective are exactly the same. This is because any net with wirelength of zero is not cut and any net which has wirelength of one is cut.

The top-down placement approach based on quadrisection has been shown effective [10, 7, 4]. This makes the hierarchical level containing 2×2 global bins very interesting to study (as shown in Figure 3a). At the 2×2 hierarchical level, the net-cut "seems" close to the wirelength objective. Let us denote four global bins by B_{11}, B_{12}, B_{21} and B_{22} as shown in Figure 3. If all nets are two-terminal, all cut nets connect two global bins. We denote the number of cut nets between

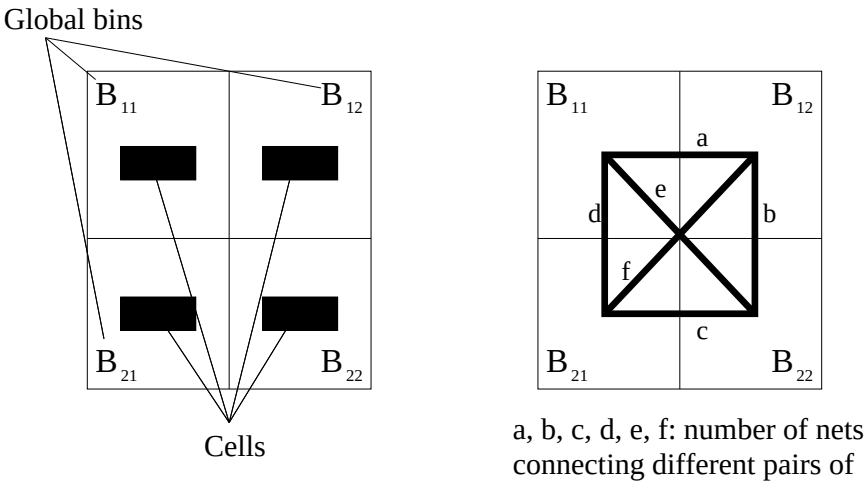


Figure 3: A hierarchical level containing 2×2 global bins.

all six possible pairs of global bins by a, b, c, d, e and f as shown in Figure 3b. Thus the total net-cut is $C_{ut} = a+b+c+d+e+f$ and the normalized wirelength $WL_n = a+b+c+d+2e+2f$. Let us compare the net-cut optimized placement and the wirelength optimized placement at this hierarchical level. In a given net-cut optimal placement (i.e., a placement that minimizes the net-cut), let the number of cut nets between global bins be a_c, b_c, c_c, d_c, e_c and f_c (note that there might be many such placement, so value of a_c, b_c, c_c,d_c, e_c and f_c is not unique),. For the wirelength optimal placement, the number of cut nets between global bins are denoted by a_w, b_w, c_w, d_w, e_w and f_w. Because the net-cut optimized placement has the smallest number of net-cut among all possible placements, we have the relation of

$$a_c + b_c + c_c + d_c + e_c + f_c \leq a_w + b_w + c_w + d_w + e_w + f_w \tag{1}$$

Similarly, the normalized wirelength of the net-cut optimized placement should be worse than the normalized wirelength of the wirelength optimized placement:

$$a_c + b_c + c_c + d_c + 2e_c + 2f_c \geq a_w + b_w + c_w + d_w + 2e_w + 2f_w \tag{2}$$

According the definition of the net-cut optimized placement, four subcircuits are placed to minimize the wirelength. Thus if we exchange the cells in B_{12} with the cells in B_{22}, we should get a worse wirelength, that is,

$$a_c + b_c + c_c + d_c + 2e_c + 2f_c \leq 2a_c + b_c + 2c_c + d_c + e_c + f_c \tag{3}$$

Similarly, when we exchange cells in B_{11} and cells in B_{12}, we establish the relation:

$$a_c + b_c + c_c + d_c + 2e_c + 2f_c \leq a_c + 2b_c + c_c + 2d_c + e_c + f_c \tag{4}$$

From (3) and (4) we have

$$e_c + f_c \leq a_c + c_c \tag{5}$$

$$e_c + f_c \leq b_c + d_c \tag{6}$$

Combining (5) and (6), we have

$$e_c + f_c \leq \frac{1}{3}(a_c + b_c + c_c + d_c + e_c + f_c) \tag{7}$$

This can be written as:

$$a_c + b_c + c_c + d_c + 2e_c + 2f_c \leq \frac{4}{3}(a_c + b_c + c_c + d_c + e_c + f_c) \tag{8}$$

When we substitue (2) into (8), we have:

$$a_w + b_w + c_w + d_w + 2e_w + 2f_w \leq \frac{4}{3}(a_c + b_c + c_c + d_c + e_c + f_c) \tag{9}$$

We also have:

$$a_w + b_w + c_w + d_w + 2e_w + 2f_w \geq a_w + b_w + c_w + d_w + e_w + f_w \geq a_c + b_c + c_c + d_c + e_c + f_c \tag{10}$$

Combining (9) and (10), we have:

$$a_c + b_c + c_c + d_c + e_c + f_c \leq a_w + b_w + c_w + d_w + 2e_w + 2f_w \leq \frac{4}{3}(a_c + b_c + c_c + d_c + e_c + f_c) \tag{11}$$

(11) shows the relationship between the net-cut cost of a net-cut optimized placement with the wirelength cost of a wirelength optimized placement. The optimal wirelength at this hierarchical level is bounded by the optimal net-cut cost. Thus the net-cut cost is a "reasonable" estimation of the real wirelength cost at 2×2 hierarchical level. The above discussion is based on the assumption that all the nets are two terminal nets. When multi-terminal nets exist, the definition of a, b, c, d, e and f changes and (3), (4) become no longer valid. However, since the number of multi-terminal nets is less than 30% (in benchmarks that we will consider later), the net-cut is still a good approximation of the wirelength cost.

Similar analysis can be performed on hierarchical levels which contain more than four global bins. Notice that the difference between the net-cut and the wirelength cost lies in the nets

which have a normalized wirelength of more than one. Based on this observation, we will define the α relations as follows: Given a particular hierarchical placement, α_i is defined as the percentage of nets which has a normalized wirelength of i. Based on this definition, the percentage of un-cut nets will be α_0 and the percentage of nets which has a normalized wirelength of one is α_1 and so on. Assume the total number of nets in the circuit is $|\mathcal{N}|$ as defined in Section 2. The net-cut for a given hierarchical placement is:

$$C_{ut} = |\mathcal{N}| \cdot (\alpha_1 + \alpha_2 + \alpha_3 + ...)$$

The normalized wirelength is:

$$WL_n = |\mathcal{N}| \cdot (\alpha_1 + 2\alpha_2 + 3\alpha_3 + ...)$$

Following the same notation used in the above analysis, the α's for the net-cut optimized placement are denoted by α_{ci} and the α's for the wirelength optimized placement are denoted by α_{wi}. Similar to (1) and (2), the optimal normalized wirelength WL_n for this hierarchical level obeys:

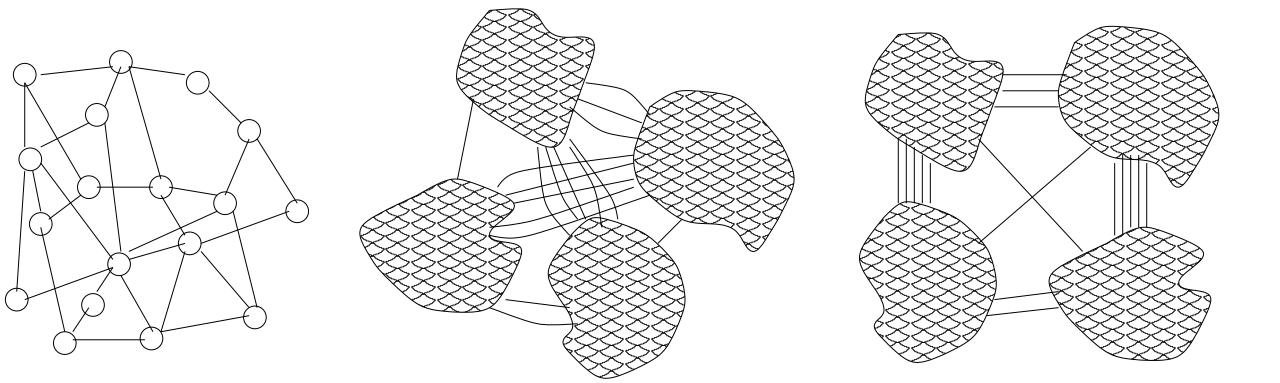
$$|\mathcal{N}| \cdot \sum_{i>1} \alpha_{ci} \leq WL_n \leq |\mathcal{N}| \cdot \sum_{i>1} i \cdot \alpha_{ci} \tag{12}$$

Equation (12) shows that the optimal wirelength can be bounded by the α's from the optimal net-cut placement. However, the quality of this bound depends on the value of α's. α's vary at different hierarchical levels. In the worst case, α_i's ($i \geq 2$) can be large and the bound can be loose. However, experiments show that α_i's ($i \geq 2$) are very small. We have to use experimental methods to determine the bound for the optimal wirelength using the net-cut optimized placement.

We experimentally get the value of α_c's of circuits in different hierarchical levels. We use four circuits from MCNC and ten from ISPD-98 benchmark suite [1] to carry out our experiments. MCNC is the standard benchmark suite for placement, but most circuits are small (less than 10,000 cells). ISPD-98 benchmark suite contains 18 large circuits constructed based on real IBM internal circuits. Table 1 shows the properties of these circuits. Since hMetis is widely claimed to be one of the best partitioners and simulated annealing is still very effective in minimizing wirelength, we use hMetis [8, 9] and simulated annealing to get the net-cut optimized placement. Figure 4 illustrate this procedure. Figure 4a shows the original circuit to be placed. We first use hMetis to partition the circuit into N_b subcircuits to minimize the net-cut where N_b is the number of global bins (Figure 4b). Then we use simulated annealing to place each subcircuit to a global bin to minimize the wirelength (Figure 4c). We get α_c's by counting the number of nets with different wirelength and divided by the total number of nets in the circuit.

Table 2 shows the α_c's for circuit Primary1. Table 3 shows the α_c's for circuit avqs. Table 4, 5, 6, 7 show the α_c's for circuits ibm01, ibm02, ibm03, and ibm04, repectively. The numbers for other circuits are very similar to the numbers shown here. Table 4 etc. also shows the upper and lower bound for the optimal normalized wirelength at each hierarchical level obtained from α_c's. We get the upper and the lower bound by Equation (12). In the coarser hierarchical

levels (e.g., 2×2, 4×4), the lower and the upper bound are quite close. However, at the finer hierarchical levels, the upper bound is getting further away from the lower bound. This fact suggests that the net-cut objective is not good to use any more at finer hierarchical levels.



(a) Original circuit.

(b) Use hMetis to partition the circuit and form clusters.

(c) Use annealing to place net-cut optimize clusters to reduce wirelength

Figure 4: The procedure to get a net-cut optimized placement.

CktName	# Cells	# IO Pads	# Nets	# Pins
Primary1	833	183	1266	3303
Primary2	3014	107	3817	12014
biomed	6417	97	7052	22253
avqs	21854	64	30038	84081
ibm01	12506	246	14111	50566
ibm02	19542	259	19584	81199
ibm03	22853	283	27401	93573
ibm04	27220	287	31970	105859
ibm05	28146	1201	28446	126308
ibm06	32332	165	34826	128182
ibm09	53110	285	60902	222088
ibm12	70439	637	77240	317760

Table 1: Properties of testing circuits.

4 How To Use a Cut-based Algorithm to Optimize Wirelength

In the hierarchical placement problem, wirelength is the objective we want to optimize. However, a net-cut optimized placement is of interest. This is because:

α_{ci}	2×2	4×4	8×8	16×16
α_{c0}	87.5%	75.0%	60.7%	43.7%
α_{c1}	10.5%	13.2%	17.8%	19.1%
α_{c2}	2.0%	6.2%	9.4%	11.0%
α_{c3}	0	4.4%	3.6%	4.8%
α_{c4}	0	1.0%	2.5%	4.3%
α_{c5}	0	0.08%	2.4%	4.1%
α_{c6}	0	0.08%	1.7%	2.7%
$\sum_{i>6} \alpha_{ci}$	0	0.0%	1.9%	10.3%
lower bound of WL_n	133	263	435	657
upper bound of WL_n	183	554	1205	2848

Table 2: Value of α_{ci} and the lower/upper bound for the optimal wirelength for circuit Primary1.

α_{ci}	2×2	4×4	8×8	16×16	32×32
α_{c0}	99.1%	98.0%	96.1%	93.2%	86.3%
α_{c1}	0.60%	1.1%	1.5%	2.3%	5.4%
α_{c2}	0.29%	0.42%	1.0%	1.5%	2.4%
α_{c3}	0	0.24%	0.48%	0.89%	1.2%
α_{c4}	0	0.13%	0.33%	0.59%	0.81%
α_{c5}	0	0.08%	0.20%	0.33%	0.59%
α_{c6}	0	0.10%	0.11%	0.18%	0.40%
$\sum_{i>6} \alpha_{ci}$	0	0.0%	0.28%	1.0%	2.9%
lower bound of WL_n	248	566	1133	1993	4081
upper bound of WL_n	353	1232	3219	7932	20990

Table 3: Value of α_{ci} and the lower/upper bound for the optimal wirelength for circuit avqs.

α_{ci}	2×2	4×4	8×8	16×16	32×32
α_{c0}	96.4%	91.0%	84.5%	69.6%	59.2%
α_{c1}	3.1%	4.6%	4.8%	11.0%	7.1%
α_{c2}	0.3%	2.4%	3.2%	4.3%	9.5%
α_{c3}	0	1.2%	2.7%	3.0%	3.6%
α_{c4}	0	0.43%	1.8%	2.4%	2.3%
α_{c5}	0	0.063%	1.1%	1.8%	1.6%
α_{c6}	0	0.070%	0.58%	1.1%	1.1%
$\sum_{i>6} \alpha_{ci}$	0	0.24%	1.32%	6.8%	15.6%
lower bound of WL_n	500	1224	2146	3529	5046
upper bound of WL_n	553	2224	6201	17588	43213

Table 4: Value of α_{ci} and the lower/upper bound for the optimal wirelength for circuit ibm01.

α_{ci}	2×2	4×4	8×8	16×16	32×32
α_{c0}	95.4%	82.1%	73.0%	65.3%	58.9%
α_{c1}	3.7%	10.1%	5.2%	3.6%	3.9%
α_{c2}	0.82%	4.3%	4.3%	3.2%	2.3%
α_{c3}	0	1.6%	5.6%	3.5%	2.4%
α_{c4}	0	1.2%	4.2%	2.6%	2.1%
α_{c5}	0	0.46%	2.6%	2.4%	1.9%
α_{c6}	0	0.097%	2.0%	2.4%	1.8%
$\sum_{i>6} \alpha_{ci}$	0	0.15%	3.1%	17%	26.7%
lower bound of WL_n	669	3272	5059	6597	7713
upper bound of WL_n	1050	6167	19042	46476	108303

Table 5: Value of α_{ci} and the lower/upper bound for the optimal wirelength for circuit ibm02.

α_{ci}	2×2	4×4	8×8	16×16	32×32
α_{c0}	92.1%	86.9%	81.4%	72.2%	62.8%
α_{c1}	5.9%	3.7%	3.6%	6.8%	6.0%
α_{c2}	1.9%	3.6%	3.5%	2.9%	6.0%
α_{c3}	0	3.6%	2.8%	2.7%	2.6%
α_{c4}	0	1.1%	2.1%	1.7%	1.8%
α_{c5}	0	0.57%	1.9%	1.6%	1.6%
α_{c6}	0	0.30%	1.2%	1.6%	1.4%
$\sum_{i>6} \alpha_{ci}$	0	0.23%	3.5%	10.5%	17.8%
lower bound of WL_n	1694	3149	4667	6685	9368
upper bound of WL_n	2677	8523	20105	49244	113653

Table 6: Value of α_{ci} and the lower/upper bound for the optimal wirelength for circuit ibm03.

α_{ci}	2×2	4×4	8×8	16×16	32×32
α_{c0}	92.3%	86.0%	79.8%	70.9%	60.4%
α_{c1}	7.0%	5.3%	5.1%	5.3%	7.0%
α_{c2}	0.70%	4.3%	4.0%	3.6%	5.2%
α_{c3}	0	3.7%	3.6%	2.9%	3.2%
α_{c4}	0	0.49%	2.6%	2.4%	2.9%
α_{c5}	0	0.16%	1.8%	2.3%	1.7%
α_{c6}	0	0.05%	0.88%	1.6%	1.3%
$\sum_{i>6} \alpha_{ci}$	0	0	2.22%	11.0%	18.3%
lower bound of WL_n	1778	3818	5951	8711	11650
upper bound of WL_n	2687	8966	21143	57650	124392

Table 7: Value of α_{ci} and the lower/upper bound for the optimal wirelength for circuit ibm04.

1. The net-cut objective is easier to optimize than the wirelength objective.

2. The net-cut objective is correlated to the wirelength objective.

3. There are a number of very good net-cut optimization packages we can use.

Therefore, it is very important to understand how to use a cut-based algorithm to optimize wirelength. First we are going to see the difference between the net-cut and the wirelength objective. Since both the net-cut and the wirelength objective cost the internal nets as zero, the difference between these two objectives lies on the cost of the external nets. We calculated the average cost of all the external nets. For the net-cut objective, this value is always 1 because every cut net has a cost of 1. Thus by looking at the value for the average normalized wirelength cost, we can see how far it is from the net-cut cost. Table 8 shows the average normalized wirelength value for the external nets in different hierarchical levels.

Circuits	2×2	4×4	8×8	16×16	32×32
Primary1	1.331	2.261	3.663	5.387	-
Primary2	1.315	2.445	4.230	6.422	9.606
biomed	1.401	2.511	4.405	6.083	9.210
avqs	1.327	2.026	2.730	3.901	5.088
ibm01	1.098	1.681	2.910	4.303	7.747
ibm02	1.177	1.673	3.245	7.048	13.052
ibm03	1.239	2.295	4.111	6.577	10.982
ibm04	1.162	2.063	3.428	5.971	10.214
ibm05	1.347	2.850	5.644	9.973	18.181
ibm06	1.424	2.308	3.880	6.189	10.914
ibm09	1.568	3.563	7.358	14.925	29.863
ibm12	1.591	3.697	7.690	15.546	31.138

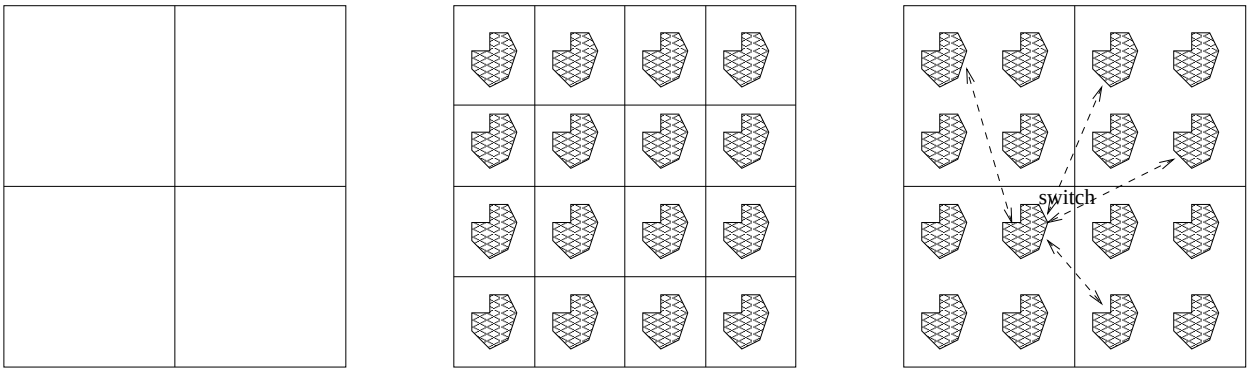
Table 8: Average normalized wirelength for all the external nets.

From Table 8, we can see that the average normalized wirelength for finer bins (above 8×8) is much larger than 1. This shows that the net-cut objective is a reasonable approximation for the wirelength in the coarser hierarchical levels but gets further away from the wirelength in the finer hierarchical levels.

4.1 An Algorithm For Combining Net-cut and Wirelength Objective

Based on the observations above, we believe a combination between the net-cut and the wirelength objective will be very effective in the hierarchical placement (as is currently done in several commercial packages). At a certain hierarchical level, we can first optimize net-cut. Then we further reduce wirelength based on the net-cut optimized placement. We can achieve

this by moving cells around the global bins to optimized the wirelength. However, it would be very slow and uneffective if we only move a single cell at a time. We propose a "+1 *level clustering*" technique to perform this task effectively: Given a hierarchical level h which has N_b global bins, first we solve the net-cut optimization problem at hierarchical level $h+1$ which has gN_b global bins where g is defined in Section 2 (usually $g = 2$ or 4). Based on the net-cut optimization result at level $h+1$, we have gN_b cell clusters with each cluster being the set of all the cells in one global bin at level $h+1$. Then we go back to the given hierarchical level h and do the wirelength optimization by placing these gN_b clusters into N_b global bins. Figure 5 shows an example of this +1 level clustering technique which $g = 4$ and $N_b = 4$. This technique will be much faster than the single cell moving algorithm because we only need to place gN_b objects into N_b places.



(a) The target level h.

(b) do net-cut optimization at level h+1. Cluster cells based on the net-cut result.

(c) do wirelength optimization optimization by moving clusters back at level h.

Figure 5: The +1 *level clustering* technique to improve wirelength.

The +1 level clustering technique can also be expand to be a +2 ,+3 or $+\delta$ level clustering technique. As δ increases, the run time will increase accordingly. If there is only one cell at each global bin at level $h+\delta$ ($\delta = \frac{\log|\mathcal{C}| - \log N_b}{logg}$), the $+\delta$ level clustering technique reduces to the single cell moving strategy.

Assuming we want to get a hierarchical placement with an optimized wirelength at level h which has N_b global bins, a step by step procedure for the +1 level clustering technique can be written as:

1. Obtain a net-cut optimized placement at level $h+1$.
2. Cluster cells in the same global bin at level $h+1$ together. There are gN_b clusters in total.
3. Do cluster placement at level h to minimize wirelength using the clusters obtained in Step 2.

We can also have a +0 level clustering technique which is to do the clustering and the wirelength optimization at the same hierarchical level h. The hierarchical placement obtained from the +0 level clustering technique is just a net-cut optimized placement since it has the optimal net-cut cost. We call this +0 level clustering technique the flat clustering technique at level h since it does not go to level $h+1$.

In the +1 level clustering technique, we use hMetis [8] as the tool to get the net-cut optimized placement. Benchmark results showed that hMetis performs really well on large sized circuits. HMetis can also handle multi-way partitioning easily. By using hMetis as our net-cut objective optimizer, we have three variations on the +1 level clustering technique. Assume that we are working on the hierarchical level h with N_b global bins:

1. +1 level A: We use hMetis to get the net-cut optimized cell clusters at level $h+1$. Then we perform the wirelength optimization at level h using simulated annealing (since the number of moving 'items' is small, an near optimal wirelength can be obtained by the annealing).

2. +1 level B: We use hMetis to get the net-cut optimized placement at level h. Then we use hMetis to partition the subcircuit in each global bin into g clusters. Then we perform the wirelength optimization at level h with these gN_b clusters using simulated annealing.

3. +1 level C: We use hMetis to get the net-cut optimized placement at the first hierarchical level h_1. Then we use hMetis to keep going down by splitting global bins until we reach level $h+1$. Then we do clustering at level $h+1$ and perform the wirelength optimization back at level h using simulated annealing.

When we split global bins to get from the level h_i to the level h_{i+1}, we use hMetis to do a g-way partition on the subcircuit in each global bin at level h_i. The subcircuit contains all the cells and terminals in that global bin. Nets in the subcircuit are modified so that they only contain terminals located in that global bin.

It is interesting to notice that given long enough time, the results from +1 level B should be no worse than the results from the flat clustering approach at level h.

We conduct experiments to compare the wirelength results from the wirelength optimized approach, the net-cut optimized approach (i.e. the flat clustering approach) and the three +1 level clustering approaches (+1 level A, B and C) at different hierarchical levels. For consistency and simplicity, we use simulated annealing to implement the wirelength optimization in the hierarchical placement. This wirelength optimization algorithm is implemented without using any clustering technique. We know that the quality of the results highly depend on the runtime of the simulated annealing. In order to make a fair comparison, we perform the simulated annealing twice for each circuit in the wirelength optimized approach. The fast simulated annealing (WL.fast) spends approximately the same amount of time as the net-cut optimized approach. The slow simulated annealing (WL.slow) is at least 3 times longer than the net-cut optimized approach.

We tested all the circuits on 2×2,4×4, 8×8, 16×16 and 32×32 hierarchical levels except for Primary1, Primary2 and biomed since they are quite small. For Primary1, we tested it on 2×2,4×4 and 8×8 levels. We test Primary2 and biomed on 2×2,4×4, 8×8 and 16×16

levels. Wirelength and runtime comparison for all circuits are shown in the Table 9 – 20. The unit for the runtime is second cpu time.

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	90	19	87	23	89	51	-	-	-	-
WL.slow	77	50	82	144	88	234	-	-	-	-
cut opt.	76	13	92	18	99	37	-	-	-	-
+1level A	91	2.7	107	4.3	100	12	-	-	-	-
+1level B	83	2.7	99	3.5	96	6.9	-	-	-	-
+1level C	83	2.7	101	3.6	97	12	-	-	-	-

Table 9: Wirelength and runtime comparison between different approaches for circuit Primary1

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	360	124	398	118	437	121	459	119	-	-
WL.slow	338	249	317	505	344	498	359	733	-	-
cut opt.	230	45	325	78	367	116	423	178	-	-
+1level A	281	11	370	15	434	27	398	75	-	-
+1level B	230	10	367	14	384	23	389	43	-	-
+1level C	256	11	349	12	499	29	396	77	-	-

Table 10: Wirelength and runtime comparison between different approaches for circuit Primary2

In a particular hierarchical level, if the wirelength optimized approach performs better than the net-cut optimized approach, it means that the net-cut objective is not a good approximation of the wirelength at this level.

1. Comparing the results of WL.fast and the results of cut.opt. in Table 9–20, we find that the net-cut optimized approach performs almost always better than the wirelength optimized approach using similar amount of time. (The only two exceptions happen at circuit ibm02 and ibm05 for level 8×8 and up.) This fact shows that the net-cut objective is a fast alternative of the wirelength.

2. Comparing the results of WL.slow and the results of cut.opt. in Table 9–20, we find a similar trend: at coarser levels, using only about one fourth of the running time, the net-cut optimized approach is still better than the wirelength optimized approach. However, the wirelength optimized approach will gradually catch up and finally becomes better than the net-cut optimized approach at a certain hierarchical level. This hierarchical level where wirelength starts to outperform net-cut is different for different circuits. For

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	272	107	332	105	345	115	358	112	-	-
WL.slow	176	4964	185	3171	146	12377	136	16400	-	-
cut opt.	73	116	102	168	142	225	168	386	-	-
+1level A	113	49	151	70	196	127	270	108	-	-
+1level B	91	49	138	51	194	75	208	98	-	-
+1level C	83	46	134	56	190	193	215	110	-	-

Table 11: Wirelength and runtime comparison between different approaches for circuit biomed

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	1325	814	1472	716	1452	742	1472	724	1581	654
WL.slow	1077	1672	1173	1739	1064	1578	1350	1712	905	3241
cut opt.	225	355	319	363	409	411	496	424	656	533
+1level A	272	219	486	186	763	207	1017	260	1456	280
+1level B	310	217	406	179	516	196	656	215	765	262
+1level C	253	212	405	188	495	179	699	227	778	265

Table 12: Wirelength and runtime comparison between different approaches for circuit avqs

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	1472	501	1517	542	2125	511	3588	490	3505	542
WL.slow	629	15709	703	16535	972	3834	879	14962	924	14846
cut opt.	384	499	596	542	847	668	1082	961	1339	1453
+1level A	409	262	737	244	1047	300	1313	384	1739	561
+1level B	385	281	649	244	876	282	1051	326	1204	524
+1level C	384	254	790	251	858	267	1028	323	1436	463

Table 13: Wirelength and runtime comparison between different approaches for circuit ibm01

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	3478	537	5080	1558	6031	1045	3659	4600	4849	4243
WL.slow	1835	30112	2475	18427	3428	6994	3287	17919	3685	16764
cut opt.	1801	836	2716	1150	3795	1895	5461	2317	6385	2607
+1**level A**	1914	541	3153	521	4850	688	5670	823	6989	1064
+1**level B**	1801	519	2704	487	3710	539	4864	711	5939	969
+1**level C**	1856	538	2580	532	3451	573	3958	638	4998	841

Table 14: Wirelength and runtime comparison between different approaches for circuit ibm02

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	6801	511	8919	1303	10070	696	12122	602	8011	2741
WL.slow	4775	2609	5472	9881	5188	9039	5397	20080	5438	32741
cut opt.	4090	591	5046	720	5420	1307	5745	1361	6269	1936
+1**level A**	4270	351	5110	365	5720	409	5987	517	6652	740
+1**level B**	4090	352	5089	327	5394	381	5556	466	5905	637
+1**level C**	4273	374	5053	366	5335	392	6036	448	7148	572

Table 15: Wirelength and runtime comparison between different approaches for circuit ibm03

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	8635	570	11537	2433	12917	1460	15138	1065	15103	1080
WL.slow	6418	3921	8139	9088	6946	9344	7008	23068	7292	17156
cut opt.	5749	1377	6730	1409	7190	1658	7670	2552	8297	2276
+1**level A**	5816	884	6806	751	7731	824	7976	986	8820	1365
+1**level B**	5762	886	6714	700	7241	766	7436	846	7939	1123
+1**level C**	5789	882	6979	801	7437	754	7752	790	9543	1036

Table 16: Wirelength and runtime comparison between different approaches for circuit ibm04

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	6357	881	8140	993	4079	3240	4220	6090	6463	5818
WL.slow	3354	23859	3787	11275	3790	5687	3939	22935	4218	21574
cut opt.	2116	1527	3507	1998	4424	2912	4670	4589	4803	6985
+1level A	2281	1115	3986	1055	4326	1381	4657	1806	5006	2250
+1level B	2116	1044	3513	953	4384	1235	4556	1542	4533	2010
+1level C	2136	1075	3836	914	4547	997	4612	1149	6324	1645

Table 17: Wirelength and runtime comparison between different approaches for circuit ibm05

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	7622	789	11686	1517	5818	2880	6246	4364	7849	4062
WL.slow	3751	16087	4550	18404	5705	6091	4773	16969	6394	15926
cut opt.	3176	1513	3854	1771	4483	2382	5482	3448	6277	4966
+1level A	3175	1232	4146	980	4958	1118	5965	1298	6846	1821
+1level B	3176	1215	4061	991	4536	1078	5413	1152	6133	1492
+1level C	3203	1178	4552	1089	5127	990	5606	1123	7784	1443

Table 18: Wirelength and runtime comparison between different approaches for circuit ibm06

	2×2		4×4		8×8		16×16		32×32	
technique	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	11481	3057	22078	3330	12045	9816	19553	3517	16449	7504
WL.slow	10123	17389	11994	24894	11204	17591	12025	20662	11591	39910
cut opt.	8335	3535	9359	2492	11005	3208	12256	4491	14620	5323
+1level A	8871	2349	11272	1812	14476	1740	16760	2034	20612	2766
+1level B	8414	2368	10335	1592	11612	1614	12422	1771	13713	2187
+1level C	8407	2335	10070	1645	11594	1626	12287	1814	13616	2168

Table 19: Wirelength and runtime comparison between different approaches for circuit ibm09

technique	2×2		4×4		8×8		16×16		32×32	
	WL	runtime	WL	runtime	WL	runtime	WL	runtime	WL	runtime
WL.fast	72622	2273	88904	12799	90574	8500	78555	10066	81028	12050
WL.slow	55031	68560	59501	40593	64548	15471	61236	33458	63602	45627
cut opt.	54878	4055	59154	4429	62037	5651	65506	8988	68071	10700
+1level A	57348	4049	62674	2916	66580	2987	70528	3662	73478	4806
+1level B	56703	4024	60197	2923	61299	2667	62962	3088	64918	3924
+1level C	56269	3965	60075	2818	61842	2708	62891	3033	64520	3585

Table 20: Wirelength and runtime comparison between different approaches for circuit ibm12

biomed and avqs (Table 11 and Table 12), wirelength does not outperform net-cut for all the levels we tested. For ibm02 and ibm05 (Table 14 and Table 17), wirelength quickly outperforms net-cut at around level 4×4. For other circuits, wirelength start outperforming net-cut at level around 8×8 or 16×16. This experiment shows that the net-cut objective is only good at coarser levels and we have to start consider wirelength at some point.

3. From Table 9–20, we see that the +1 level approaches are no better than the net-cut approach at coarser levels. The +1 level approach start to outperform the net-cut approach at finer levels. By using less amount of time, the +1 level approach produces better wirelength results than the net-cut optimized approach. We know considering wirelength is necessary at finer levels. However, traditional wirelength optimized approach needs very long time to get good results. The +1 level approach effectively combines net-cut and wirelength together. Thus the +1 level approach is much faster than the pure wirelength approach since it uses net-cut to cluster cells. The hierarchical placement produced by the net-cut approach is always a solution favoring the net-cut cost. In finer hierarchical levels, this solution will no longer be close to the wirelength optimal solution. Since the +1 level approach takes the wirelength into account, it has better performance than the net-cut approach in finer levels. Of the three approaches (A, B and C) for the +1 level clustering technique, experiments show that approach A is always bad and approach B is the best one.

Based on the discussion above, we conclude that it is wise to use the net-cut objective at early hierarchical levels and start considering wirelength at later levels. This is consistent with the analysis we made in the previous section.

4.2 Where to Switch from Net-cut to Wirelength

The previous experiments show that wirelength needs to be considered at later hierarchical levels. However, yet there is no clue when is a good point to start considering wirelength (using the +1 level clustering heuristics). The difference between the net-cut and wirelength objective is in the cost of external nets. The more the external nets, the bigger the difference is. We

believe that we can tell when to start considering wirelength by looking at the percentage of the external nets. We plot the curve of the percentage of the external nets vs. the number of global bins in the hierarchical level. Figure 6 and Figure 7 show curves for all testing circuits. The solid curve with '+' on it is the curve of external nets percentage vs. the number of global bins. Other five curves on each sub-figure represent a theoretical curve derived from Rent's Rule with different Rent parameters.

Rent's rule was first described by Landman and Russo in 1971 [11]. When we partition a circuit into several blocks, Rent's rule is about the relationship between the number of external pins on each block and the number of cells inside each block. Let us denote the average number of cells in a block by B_m, then the average number of external pins P_m can be expressed as $P_m = T_b B_m^r$ where $0 \leq r \leq 1$ is called the rent parameter and T_b is the average number of terminals per block. Rent's rule is experimentally validated for a lot of real circuits and for different partitioning methodologies. For real circuits, the Rent parameter r usually has a value of between 0.3 and 0.8. If a circuit obeys Rent's rule, we can derive a theoretical relationship between the external nets percentage and the number of global bins. Assume we have N_b global bins with all the cells distributed in them evenly, thus we have $\frac{|\mathcal{C}|}{N_b}$ cells in each global bin. According to Rent's rule, the number of external pins on each global bin $P_m = T_b(\frac{|\mathcal{C}|}{N_b})^r$. The total number of external nets will be $\frac{P_m}{p_{avg}} N_b = N_b \frac{T_b}{p_{avg}} (\frac{|\mathcal{C}|}{N_b})^r$ where p_{avg} is the average number of terminals per net. Thus the percentage of the external nets is $p_{ext} = \frac{N_b}{|\mathcal{N}|} \frac{T_b}{p_{avg}} (\frac{|\mathcal{C}|}{N_b})^r$. To get the value of T_b, we know that the percentage is 1 when we have $\mathcal{C}$ global bins with one cell in each bin. Thus we have relation:

$$1 = \frac{|\mathcal{C}|}{|\mathcal{N}|} \frac{T_b}{p_{avg}} (\frac{|\mathcal{C}|}{|\mathcal{C}|})^r = \frac{|\mathcal{C}|}{|\mathcal{N}|} \frac{T_b}{p_{avg}} \tag{13}$$

Which gives us the value of $T_b = \frac{p_{avg}|\mathcal{N}|}{|\mathcal{C}|}$. Plug this value back into the relation, we have $p_{ext} = (\frac{|\mathcal{C}|}{N_b})^{r-1}$. Since the actual value of the Rent parameter r varies from circuit to circuit, for each circuit, we plot five theoretical curves each one with a different Rent parameter value ranging from 0.3–0.7. Figure 6 shows the curves for four MCNC benchmark circuits (Primary1, Primary2, biomed and avqs). Figure 7 shows the curves for circuit ibm01 – ibm06. Figure 6 and 7 show that Rent's curve is not exactly obeyed by the real circuits. It is usual that the circuit has different values of Rent parameter in different hierarchical levels. This is actually consistent with the way to design VLSI circuits. The hierarchical design methodology in VLSI tends to combine a number of small subcircuit into one big circuit. However, the degree of complexities is different according to the size of the circuit. When the size of a circuit is small, it is possible to put very complicated logic in it. Thus it will have a larger Rent parameter. When the size of a circuit is large, the logic between subcircuits will be comparably simple. Thus it will have a smaller Rent parameter. As shown in Figure 6 and 7, most circuit has a smaller Rent parameter in early hierarchical levels (large subcircuits) than in later levels (small subcircuit).

The difference between the net-cut and the wirelength objective are the cost on external nets. Thus the more the external nets, the bigger difference there is between these two objectives.

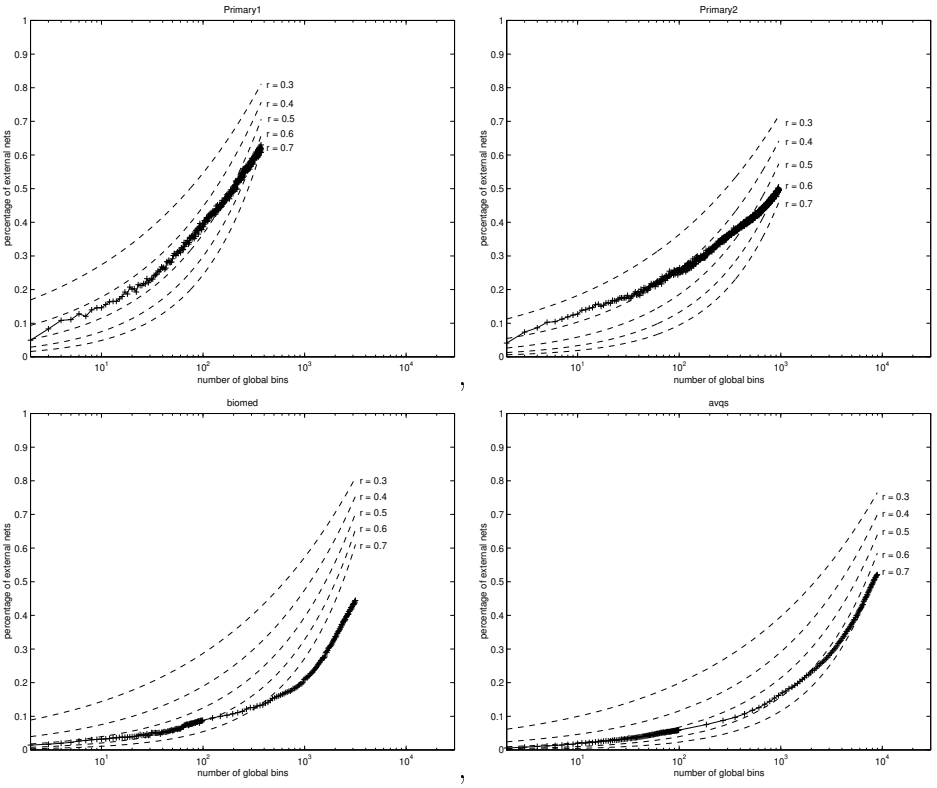


Figure 6: Percentage of external nets vs. number of global bins.

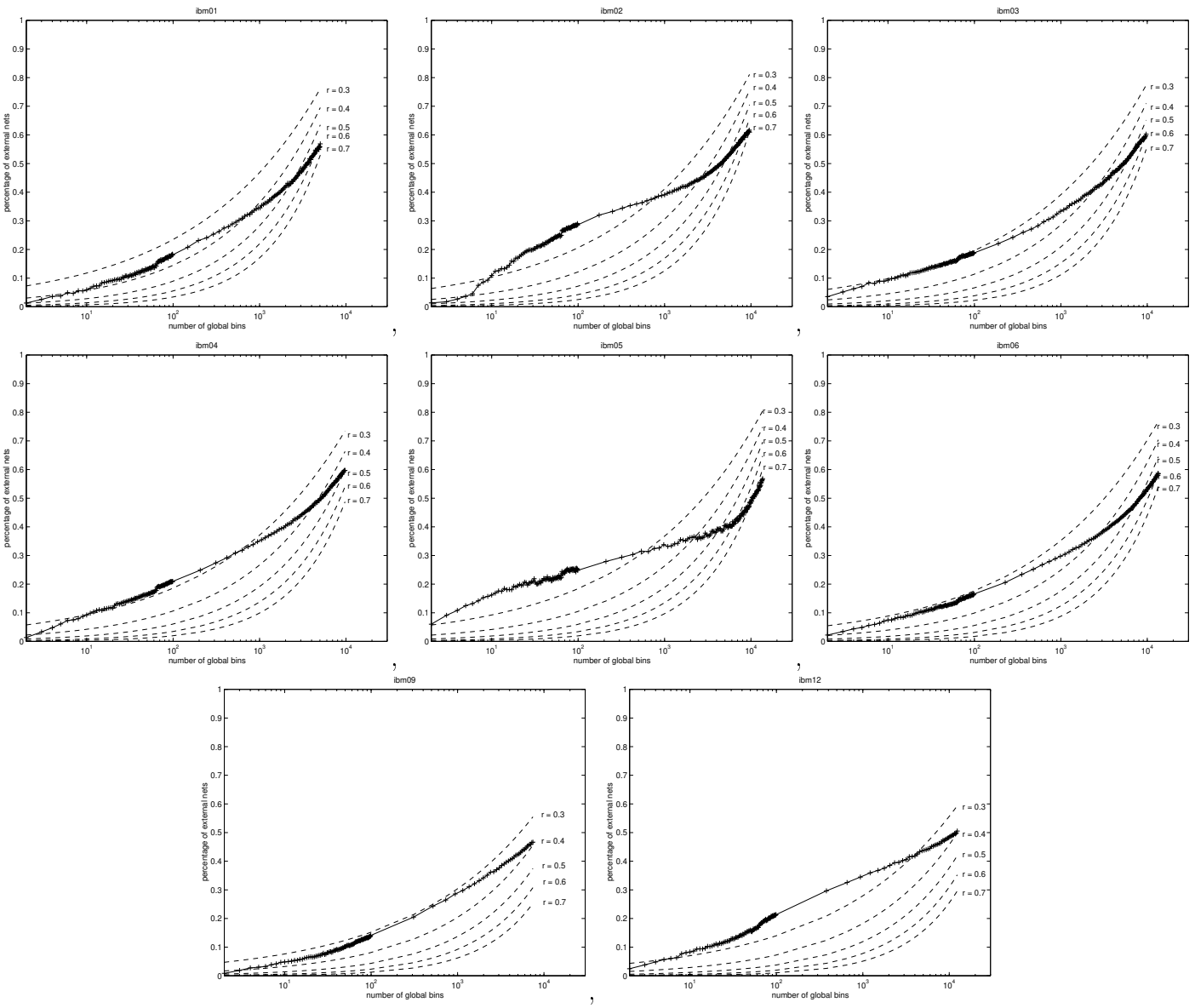


Figure 7: Percentage of external nets vs. number of global bins.

Based on the percentage curves of external nets and the experimental data in Table 9 – 20, we empirically found that $20\% - 30\%$ is the percentage where we should start considering wirelength. When less than $20\% - 30\%$ nets are external in a hierarchical level, net-cut is a very reasonable estimation of wirelength. Thus we can use net-cut objective at this hierarchical level. If more than $20\% - 30\%$ nets are external, net-cut is no longer a good objective to use to minimize wirelength. We should start using wirelength as the optimization objective. For instance, based on the curve for circuit Primary2 in Figure 6, 25% external net ratio is corresponding to about 100 global bins. From Table 10, we find that the wirelength objective starts to outperform the net-cut objective between 4×4 and 8×8 bins. In Figure 7, for circuit ibm01, 25% external net ratio is corresponding to about 200 global bins. From Table 13, we find that the wirelength objective outperforms the net-cut objective at 16×16 bins. In fact, most circuits have the 25% external net ratio at about 100 – 500 global bins. Experimental results show that the wirelength of these circuits outforms the net-cut after 8×8 or 16×16 bins. Two MCNC benchmark circuits, biomed and avqs, have the 25% external net ratio at more than 1000 global bins (Figure 6). Correspondingly, Table 11 and 12 show that the net-cut approach is always the best up to level 32×32. When few external nets are exposed, net-cut is always a good alternative of wirelength. This "$20\% - 30\%$ external nets" rule is based on the intuition and the actual experimental results. It is an approximate rule. Constructing a external net ratio curve for a circuit is very time consuming. However, we do not really need the whole curve to determine the place where we need to switch to wirelength. At a given hierarchical level, we can decide whether we should consider wirelength by looking at the external net ratio at this level. Thus it is very easy and convenient to make the decision based on the net-cut result.

5 Conclusion

In this paper we defined α's to express the difference between wirelength and net-cut at different hierarchical levels. We showed that the net-cut objective is a good approximation of length at coarser hierarchical levels. At finer levels the net-cut objective gets further from the wirelength objective. Experimental results are shown to support this claim. Based on this conclusion, a good way to minimize wirelength for a top-down approach is to consider net-cut at early hierarchical levels and switch to wirelength later. We proposed a "+1 level clustering" technique. Experiments show that this technique can effectively combine the advantage of minimizing net-cut (fast) and wirelength (accurate) together in later hierarchical levels when we start considering wirelength. Finally we showed that the percentage of external nets is important to determine where we should switch from the net-cut objective to the wirelength objective. Experimental data showed that if more than $20\% - 30\%$ nets are external, wirelength should be considered in the optimization objective. Otherwise, net-cut is a reasonable estimation of wirelength.

References

[1] C. Alpert. "The ISPD98 Circuit Benchmark Suite". In *International Symposium on Physical Design*, pages 18–25. ACM, April 1998.

[2] C. J. Alpert, J. H. Huang, and A. B. Kahng. "Multilevel Circuit Partitioning". In *Design Automation Conference*, pages 530–533. IEEE/ACM, 1997.

[3] J. Cong, H. P. Li, S. K. Lim, T. Shibuya, and D. Xu. "Large Scale Circuit Partitioning with Loose/Stable Net Removal and Signal Flow Based Clustering". In *International Conference on Computer-Aided Design*, pages 441–446. IEEE, 1997.

[4] A. E. Dunlop and B. W. Kernighan. "A Procedure for Placement of Standard Cell VLSI Circuits". *IEEE Transactions on Computer Aided Design*, 4(1):92–98, January 1985.

[5] D. Dutt and W. Deng. "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques". In *International Conference on Computer-Aided Design*, pages 194–200. IEEE, 1996.

[6] S. Hauck and G. Boriello. "An Evaluation of Bipartitioning Techniques". In *Chapel Hill Conference on Advanced Research in VLSI*, 1995.

[7] D. Huang and A. B. Kahng. "Partitioning-based Standard-cell Global Placement with an Exact Objective". In *International Symposium on Physical Design*, pages 18–25. ACM, April 1997.

[8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. "Multilevel Hypergraph Partitioning: Application in VLSI Domain". In *Design Automation Conference*, pages 526–529. IEEE/ACM, 1997.

[9] G. Karypis and V. Kumar. "Multilevel k-way Hypergraph Partitioning". In *Design Automation Conference*, pages 343–348, 1999.

[10] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization". *IEEE Transactions on Computer Aided Design*, 10(3):365–365, 1991.

[11] B. Landman and R. Russo. " On a pin versus block relationship for partitions of logic graphs". *IEEE Transactions on Computers*, c-20:1469–1479, 1971.

[12] M. Sarrafzadeh and M. Wang. "NRG: Global and Detailed Placement". In *International Conference on Computer-Aided Design*. IEEE, November 1997.

[13] C. Sechen. *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer, B. V., Deventer, The Netherlands, 1988.

[14] C. Sechen and K. W. Lee. "An Improved Simulated Annealing Algorithm for Row–Based Placement". In *Design Automation Conference*, pages 180–183. IEEE/ACM, 1988.

[15] G. Sigl, K. Doll, and F. M. Johannes. "Analytical Placement: A Linear or a Quadratic Objective Function". In *Design Automation Conference*, pages 427–431. IEEE/ACM, 1991.

[16] M. Wang and M. Sarrafzadeh. "Behavior of Congestion Minimization During Placement". In *International Symposium on Physical Design*, pages 145–150. ACM, April 1999.