A Gradient Method on the Initial Partition of Fiduccia-Mattheyses Algorithm

Lung-Tien Liu[+], Ming-Ter Kuo, Shih-Chen Huang[++], and Chung-Kuan Cheng

[+]AT&T Bell Laboratories
Murray Hill, NJ07974

Computer Science and Engineering
University of California, San Diego

[++]Department of Computer Science
New York University

Abstract— In this paper, a Fiduccia-Mattheyses (FM) algorithm incorporating a novel initial partition generating method is proposed. The proposed algorithm applies to both bipartitioning and multi-way partitioning problems with or without replication. The initial partition generating method is based on a gradient decent algorithm. On partitioning without replication, our algorithm achieves an average of 17% improvement over the analytical method, *PARABOLI*, on bipartitioning, 10% better than *Primal – Dual* method on 4-way partitioning and 51% better than net-based method. On partitioning allowing replication, our algorithm achieves an average of 23% improvement over the directed Fiduccia-Mattheyses algorithm on Replication Graph (FMRG) method on bipartitioning.

1 Introduction

Many approaches [1, 3, 4, 5, 6, 7, 9, 14, 17, 18, 19, 20] have been proposed to solve the partitioning problem with size constraints. Among them, the Kernighan-Lin [9] based algorithms are most widely used due to its simplicity. An efficient variant was later proposed by Fiduccia and Mattheyses [6]. Sanchis [15] adapted the two-way Fiduccia-Mattheyses (FM) algorithm to deal with multi-way partitioning problems.

In practice, a given cell can be replicated to reduce the inter-chip connection. Kring and Newton [10] extended the FM algorithm to allow nodes to be duplicated explicitly during partitioning. Hwang and El Gamal [8] formulated the replication problem as a problem to determine optimum replication sets for an existing partitioning. Later, Liu et. al. [12] devised a replication graph to consider the combined effect of partitioning and replication. Then, a directed FM algorithm was applied on the proposed replication graph.

FM based algorithms are highly sensitive to the choices of the initial partition. Therefore, a good initial partition is critical for the success of an FM based algorithm. In this paper, we utilize a gradient decent algorithm [16] to generate an initial partition of the FM based algorithm. Given a circuit, we formulate the partitioning problem (with or without replication) as an integer mathematical programming problem with quadratic objective function and linear constraints. By removing the constraints, the partitioning problem becomes an unconstrained minimization problem. Then, a gradient decent approach is applied to solve the unconstrained problem. Based on the solution generated by the gradient decent algorithm, we construct a feasible solution of the partitioning problem (i.e., satisfying all constraints) by solving an assignment problem. Finally, an FM based algorithm is applied to improve the solution.

2 Problem Statements

2.1 Models and Definitions

The directions of nets have to be imposed in partitioning with replication. Therefore, we introduce a directed hypergraph and a directed graph to model a circuit.

Directed Hypergraph Model: Given a circuit, we formulate its netlist as a directed hypergraph $\widehat{H} = (V, \widehat{E}^H)$, where set V consists of nodes i $(i = 1, \ldots, n)$ with size s_i and set $\widehat{E}^H$ consists of nets e_u $(u = 1, \ldots, m)$ with weight c_u. A multiple-pin net e_u is characterized by (a_u, b_u) where $a_u \subset V$ are the source nodes and $b_u \subset V$ are the sink nodes. We assume that $|a_u \cup b_u| \geq 2$, $|a_u| \geq 1$ and $|b_u| \geq 1$.

Directed Cut Set: For two disjoint node sets X and Y, we use $(X \leftarrow Y)$ to denote the directed cutset from Y to X. Therefore, $(X \leftarrow Y)$ contains all the nets $e_u = (a_u, b_u)$ such that Y intersects source node set a_u and X intersects sink node set b_u. We use function $c(X \leftarrow Y)$ to denote the total weight of the nets in $(X \leftarrow Y)$.

Directed Graph Model: Given a hypergraph, hyperedges in the netlist can be transformed into graph edges based on a complete bipartite graph model, where each directed net $e_u = (a_u, b_u)$ contributes a set of graph edges $\{(i, j) \mid i \in a_u,\ j \in b_u\}$ with weight $\frac{c_u}{|b_u|}$ on each edge (i, j). Given a directed hypergraph $\widehat{H} = (V, \widehat{E}^H)$, let $\widehat{G} = (V, \widehat{E})$ denote the corresponding directed graph of $\widehat{H}$, where $\widehat{E}$ consists of edges (i, j) with weight $c_{i,j}$.

The graph model is used to simplify the derivation of the gradient function. Once the gradient function is formulated, we extend the derivation to the hypergraph model.

Multi-Way Partitioning: Let K denote the number of blocks. Given a set of nodes V, a K-way partition $P = (V_1, V_2, \ldots, V_K)$ maps V into K blocks, such that $V_1 \cup V_2 \cup \ldots \cup V_K = V$. We use function $size(X)$ to denote the total size of a node set X. The upper and lower size limits of each partition are denoted by C_U and C_L, respectively. C_L and C_U are defined as follows:

$$C_L = \frac{size(V)}{K}(1 - \frac{\alpha}{100}), \qquad C_U = \frac{size(V)}{K}(1 + \frac{\alpha}{100}),$$

where α is a user-specified parameter to control the size constraints. We call α the size overhead parameter.

2.2 Objective Functions of Multi-Way Partitioning Problem

There are four different objective functions used in the partitioning literature.
Obj 1: Minimize the maximum number of I/O pins of each blocks.
Obj 2: Minimize the number of crossing nets.
Obj 3: Minimize the total number of I/O pins on all blocks.
Obj 4: Minimize the total number of input pins on all blocks.

In this paper, the gradient decent approach uses Obj 4 as an objective function to derive the initial partition. Once the initial partition is derived, we adopt the FM method to improve the result using the objective function specified by the user.

2.3 Problem Formulation

K-way Partitioning Problem without Replication: *Given a directed hypergraph $\widehat{H} = (V, \widehat{E}^H)$, find a partition $P = (V_1, V_2, \ldots, V_K)$ with an objective of minimizing the total number of input pins, i.e.* $\min \sum_{1 \le i \le K} c(V_i \leftarrow \bar{V}_i)$, *subject to the size constraints,* $C_L \le size(V_i) \le C_U \quad \forall$ *block* $i \in \{1, 2, \ldots, K\}$, *the feasible condition,* $V_1 \cup V_2 \cup \ldots \cup V_K = V$, *and the disjoint condition,* $V_i \cap V_j = \emptyset \quad \forall i, j \in \{1, 2, \ldots, K\}$ *and* $i \neq j$.
The feasible condition expresses that all nodes should be assigned into blocks. When replication is not allowed, the disjoint condition restricts each node to be placed into only one block.

K-way Partitioning Problem with Replication: The definition of the K-way partitioning problem with replication is the same as that of the partitioning problem without replication except the disjoint condition is excluded.

Although replication may change the netlist of the original circuit, the objective function above remains valid for the problem.

Lemma 1: *The objective function* $\sum_{1 \le i \le K} c(V_i \leftarrow \bar{V}_i)$ *represents the total number of input pins on all blocks after replication.*

3 Heuristic Algorithm for Partitioning Problems

In this section, we first formulate the partitioning problems as integer mathematical programming problems. Then, we outline the heuristic algorithm based on the gradient decent approach to tackle the partitioning problems. Finally, a two-phase partitioning method is introduced.

3.1 Integer Mathematical Programming Formulation

We use a vector of boolean variables to represent a multi-way partition so the partitioning problems can be formulated as integer mathematical programming problems. For simplicity, we formulate the partitioning problems based on the graph model. The extension to the hypergraph model will follow.

Boolean Vector: For a node i and a block b, let $x_{b,i}$ denote a boolean variable, where $x_{b,i}$ is 1 if node i is assigned to block b, otherwise $x_{b,i}$ is 0. Then a vector $x = (x_{1,1}, \ldots, x_{1,n}, \ldots, x_{K,1}, \ldots, x_{K,n})$ can present a K-way partition.

Partitioning without Replication: Given a directed graph $\widehat{G} = (V, \widehat{E})$, the partitioning problem is formulated as follows:

$$\min \ f(x) \ = \sum_{1 \le b \le K} \sum_{1 \le i \le n} \sum_{(j,i) \in \widehat{E}} c_{j,i} x_{b,i} (1 - x_{b,j}) \quad (1)$$

Subject to the following constraints:

$$C_L \le \sum_{i=1}^{n} x_{b,i} s_i \le C_U \qquad \forall \ block \ b \in \{1, 2, \ldots, K\}. \quad (2)$$

$$\sum_{b=1}^{K} x_{b,i} = 1 \qquad \forall \ node \ i \in V. \quad (3)$$

The objective function is to minimize the total number of input pins on all blocks. Constraint (2) states that the total node size of each block should be within the specified bounds. Constraint (3) states that each node is placed into only one block. In the sequel, we use $S \subseteq \{0,1\}^{Kn}$ to denote the set of feasible integer solutions satisfying equations (2) and (3).

Partitioning with Replication: The mathematical formulation of the multi-way partitioning problem with replication has the same objective function and size constraint as those of the partitioning problem without replication. However, the equality constraint (3) is replaced by an inequality expression,

$$\sum_{b=1}^{K} x_{b,i} \ge 1 \qquad \forall \ node \ i \in V. \quad (4)$$

3.2 Outline of the Heuristic Algorithm for Partitioning Problem

The gradient decent approach is first introduced, followed by the initial partition generating method based on this approach. Finally, we outline the heuristic algorithm for the partitioning problem.

3.2.1 Gradient Decent Approach

The gradient decent algorithm ([16] pp.226) can tackle the unconstrained minimization problem. Given a continuous and differentiable function f defined over R^d, the approach generates a vector sequence $\{t^h\}_{h=1}^{\infty}$ given by

$$t^{h+1} \ = \ t^h - \frac{\nabla f(t^h)}{w^h}, \quad (5)$$

where $t^h \in R^d$, $\nabla f(t^h)$ represents the first partial derivative of function f at point t^h, and w^h is a step function to normalize the search direction $\nabla f(t^h)$ from point t^h. The generated sequence will converge to a local minima of function f.

3.2.2 Initial Partition Generating Scheme

The gradient decent algorithm cannot apply to partitioning problems which are defined over an integer domain and have linear constraints. In subsection 3.1, the partitioning problems have the following generic formulation:

$$\min \quad f(x) \quad \text{s.t.} \quad x \in S \subseteq \{0,1\}^{Kn},$$

where K and n denote the numbers of partitions and nodes, respectively.

By removing the constraint $x \in S \subseteq \{0,1\}^{Kn}$ of the partitioning problems, the gradient decent algorithm will generate a sequence of points which eventually converges to a local minima.

Step Function: In equation (5), a step function is required to calculate the next point in the sequence. Given a point t^h, we set the step function equal to the value of the objective function, i.e. $w^h = f(t^h)$, to normalize the search direction.

Maximization Problem for Generating Feasible Solution: Let t^h denote a point generated by the gradient decent algorithm at the h-th step, where $t^h = (t^h_{1,1}, \ldots, t^h_{1,n}, \ldots, t^h_{K,1}, \ldots, t^h_{K,n})$. We can construct a feasible solution x^h of the partitioning problem from t^h. The intuition is that the value $t^h_{b,i}$ represents the likelihood of assigning a node i to a block b. We intend to obtain a solution x^h with the following property: x^h_i is equal to 1 if the value t^h_i is large; x^h_i is equal to 0 if t^h_i is small. Thus, given a point t^h, the following maximization problem is solved in order to generate a feasible solution of the partitioning problem:

$$\max \sum_{1 \le i \le n} x_i t^h_i \quad \text{s.t.} \quad x \in S.$$

Initial Partition Generating Scheme: Given a partitioning problem, the gradient decent approach can be applied to generate a result which may not be in the solution space S. In that case, we can produce a feasible solution by solving a maximization problem based on the infeasible result. The generated feasible solution can be further refined by adopting an iterative improvement algorithm.

3.2.3 Outline of Algorithm

We show the outline of the heuristic algorithm for tackling the partitioning problem.

1. Initially, $h = 1$; $t^0 = x^0 \in S$;
2. $t^h = t^{h-1} - \frac{\nabla f(x^{h-1})}{f(x^{h-1})}$;
3. Solve the following maximization problem to generate y^h;
   $\max \sum_{1 \le i \le n} y_i t^h_i \quad s.t. \quad y \in S$;
4. Apply an FM based method on y^h to generate a better solution x^h;
5. Record x^h, if x^h is so-far best;
6. $h = h + 1$; If $h \le Iter$, then goto 2; otherwise, stop.

Initially, we randomly choose a point from S. In each iteration, Step 2 generates a new point based on the gradient decent method. Then, a maximization problem is solved to produce a new feasible solution $y^h \in S$ in Step 3. In Step 4, an FM based iterative improvement algorithm using y^h as an initial partition is applied to generate a better result x^h. The iteration stops if the number of iterations reaches the limit.

3.3 Algorithm for Partitioning Problem without Replication

Based on the outline of the heuristic algorithm in subsection 3.2.3, the detailed algorithm for the partitioning problem without replication is presented.

Partial Derivative: We calculate the first partial derivative of the objective function in Step 2 of the algorithm outline as follows. Given a directed graph $\widehat{G} = (V, \widehat{E})$, the objective function f of the partitioning problem without replication is equal to equation (1). Given a vector of boolean values x^h, let vector z^h denote the derivative $\nabla f(x^h)$ of function f, i.e. $z^h_{b,i} = \frac{\partial f(x^h)}{\partial x_{b,i}}$. The value $z^h_{b,i}$ is equal to $\sum_{(j,i) \in \widehat{E}} c_{j,i}(1 - x^h_{b,j}) - \sum_{(i,j) \in \widehat{E}} c_{i,j} x^h_{b,j}$. Given a block b and node i, if i has input signal from node j (i.e., (j,i) is in $\widehat{E}$) and j is not in block b, the term $c_{j,i}(1 - x^h_{b,j})$ will contribute a value $c_{j,i}$ to $z^h_{b,i}$. Therefore, the term $\sum_{(j,i) \in \widehat{E}} c_{j,i}(1 - x^h_{b,j})$ denotes the total weight of the edges (j,i) of which node j is not in block b. Similarly, the term $\sum_{(i,j) \in \widehat{E}} c_{i,j} x^h_{b,j}$ represents the total weight of the edges (i,j) of which node j is in block b.

Extension to Directed Hypergraph Model: Given a directed hypergraph $\widehat{H} = (V, \widehat{E}^H)$ and a vector of boolean values x^h, the value $z^h_{b,i}$ for the directed hypergraph is defined as $c(\{e_u = (a_u, b_u) \mid i \in b_u,\ V^h_b \cap a_u \neq a_u\}) - c(\{e_u = (a_u, b_u) \mid i \in a_u,\ V^h_b \cap b_u \neq \emptyset\})$, where V^h_b denotes the set of nodes assigned to block b with respect to vector x^h.

Assignment Problem: Let t^h denote the vector generated by the gradient decent approach at the h-th iteration. The maximization problem described in Step 3 of the algorithm outline is defined as follows for the partitioning problem without replication:

$$\max \sum_{1 \le b \le K} \sum_{1 \le i \le n} x_{b,i} t^h_{b,i} \tag{6}$$

Subject to the following constraints:

$$C_L \le \sum_{i=1}^{n} x_{b,i} s_i \le C_U \qquad \forall\, block\ b \in \{1, 2, \ldots, K\}. \tag{7}$$

$$\sum_{b=1}^{K} x_{b,i} = 1 \qquad \forall \ node \ i \in V. \tag{8}$$

The above maximization problem is a classic assignment problem. Let $t^h_{b,i}$ be interpreted as the gain by assigning node i to block b. The assignment problem is to find a node assignment with a maximum gain subject to the constraints that each node is assigned to only one block and the total node size of each block is within the size bounds. The assignment problem is $\mathcal{NP}$-Hard [13]. We adopt the heuristic algorithm described in [13] to solve the assignment problem.

Multi-Way Fiduccia-Mattheyses Algorithm: We adopt the multi-way Fiduccia-Mattheyses (FM_k) algorithm [15] as the iterative improvement method for the partitioning problem without replication. In this paper, we do not incorporate the higher level gain computation in our FM_k algorithm. Note that FM_k algorithm can be adjusted to handle different objective functions as specified by the user.

3.4 Algorithm for Partitioning Problem with Replication

The heuristic algorithm for partitioning with replication has the same outline as the problem without replication. However, the assignment problem and the iterative improvement algorithm are extended for replication.

Assignment Problem Allowing Replication: The maximization problem for the partitioning problem with replication has the same objective function and size constraint as those of the partitioning problem without replication. However, we use the following constraint to replace constraint (8).

$$\sum_{b=1}^{K} x_{b,i} \geq 1 \qquad \forall \ node \ i \in V. \tag{9}$$

We call the above problem as the *Assignment Problem with Replication*, since constraint (9) allows node replication. For the solvability of the problem, we have the following two theorems.

Theorem 1: *The assignment problem with replication can be solved in polynomial time, if every node has unit size.*

Theorem 2: *The assignment problem with replication is $\mathcal{NP}$-Hard, if the problem has non-unit node size.*

Algorithm for Assignment Problem with Replication: Given n nodes and K blocks, let $t_{b,i}$ denote the gain that node i is assigned into block b. The following shows the outline of the heuristic algorithm to solve the assignment problem with replication.

1. Solve assignment problem without replication;
2. Sort list $(t_{1,1}, \ldots, t_{K,1}, \ldots, t_{1,n}, \ldots, t_{K,n})$ in descending order into $(t'_1, \ldots, t'_{Kn})$;
3. For $i = 1$ to Kn
   if ($t'_i \leq 0$) stop;
   else if (having enough space) do duplication;

We first adopt the heuristic algorithm for the assignment problem without replication to generate an initial assignment. Then, we consider additional node assignments in Step 3 for duplication in the descending order of their gains. Only those assignments with positive gains are examined since they will increase the total gain of the resulted assignment. Therefore, if the current node assignment has negative gain, the algorithm stops. Otherwise, we check if the duplication will violate the size constraint. If not, we perform the node duplication.

Directed Multi-Way Fiduccia-Mattheyses: The directed two-way Fiduccia-Mattheyses algorithm in [8] is extended to perform the multi-way partitioning with replication. We use FM_{kr} to denote the K-way Directed Fiduccia-Mattheyses algorithm.

3.5 Two-Phase Partitioning

A two-phase partitioning approach [4, 2] can reduce the computational complexity of huge circuits. Given a circuit, we first do clustering. In the first phase, the partitioning algorithm is applied to the condensed circuit induced from the clustering. Then, in the second phase, we use the expanded partition from the first phase as the starting point for the partitioning algorithm on the flat circuit. We recursively apply the two-way ratio-cut partitioning algorithm [4] to divide into highly connected group.

4 Experiments

The algorithm for the partitioning problem without replication on the flat circuit is called the *Gradient decent based Fiduccia-Mattheyses* (GFM) algorithm. We use the subscript k as in GFM_k to represent the version for multi-way partitioning. For partitioning with replication, we use subscript r as in GFM_r to denote the algorithm that allows cell replication. The subscript t as in GFM_t represents the partitioning algorithm on clustered circuits.

In the experiments, all algorithms adopted Obj 2 as the objective function. Therefore, subcolumn *cut* in each table denotes the crossing net count. Subcolumn *exe* lists the execution time on a Sun Sparc 10 and measured in seconds.

4.1 Partitioning without Replication

4.1.1 Bipartitioning

Comparison with Cheng and Wei's stable algorithm [4]: In this experiment, each cell was given the actual area size. We also followed [4] to set each I/O pads with the minimum area size of all cells. The experiment was based on the size overhead parameter 50%. Table 1 shows the experimental results. Column $RCII$ denotes the results from [4]. FM and GFM run on flat circuits. Column FM lists the best results from 1600 iterations. GFM reports the best results from the experiments with the number of iterations 60 (i.e., totally call two-way FM to perform iterative improvement 60 times during the execution of GFM) and 80, respectively. The execution time *exe* of FM and GFM reports the average running time for

each iteration. $RCII$ and GFM_t are two-phase based algorithm. $RCII$ and GFM_t report the best results from 20 runs where each run call FM 20 times on clustered circuits. The subcolumn exe in GFM_t lists the average execution time for each run. Overall, the two-phase algorithm GFM_t achieves the best improvement over $RCII$.

	RCII	FM		GFM			GFM_t	
	cut	cut	exe	cut	cut	exe	cut	exe
cir.		1600		60	80			
T4	42	44	0.7	44	44	0.7	42	19
T6	55	52	1.0	49	49	1.5	48	48
19ks	80	97	3.0	79	78	2.5	76	56
26K	65	76	149	70	70	54.8	49	593

Table 1: *Comparison with FM approach and stable algorithm.*

Comparison with analytical method PARABOLI [14]: We used the test cases from the authors of [14]. In this experiment, each cell was given an unit size. The size overhead was set to 10% of the total cell size. Table 2 reports the comparison results. We use PA to denote the analytical method $PARABOLI$. The results of GFM_t are from one run only. On the average, the GFM algorithm with the number of iterations 80 and the two-phase GFM_t algorithm achieve 15% and 17% improvements over $PARABOLI$, respectively.

	PA	FM		GFM			GFM_t	
	cut	cut	exe	cut	cut	exe	cut	exe
cir.		1600		60	80			
s1423	16	15	0.1	16	16	0.1	15	14
sioo	45	28	0.2	25	25	0.2	28	22
s1488	50	43	0.2	46	46	0.3	51	26
balu	41	27	0.5	27	27	0.3	28	25
p1	53	47	0.3	47	47	0.2	51	25
struct	40	42	1.2	43	41	1.0	36	32
p2	146	161	3.7	139	139	2.8	139	61
s9234	74	47	11	42	41	8.4	44	186
biomed	135	83	21	84	84	18	92	371
s13207	91	77	30	67	66	24	61	397
s15850	91	84	47	63	63	32	46	530
ind2	193	298	73	211	211	54	175	819
ind3	267	241	77	241	241	50	244	861
s35932	62	99	277	48	41	127	44	1088
s38584	55	59	488	50	47	121	54	3463
s38417	49	135	588	85	81	141	62	1062

Table 2: *Comparison with FM approach and analytical method PARABOLI.*

4.1.2 Multi-Way Partitioning

Comparison with *Primal-Dual* method: In this experiment, each cell was given the actual area size. The size overhead was limited to 50% of the total cell size. Table 3 shows the experimental data. The GFM_k algorithm was performed with the number of iterations 40. Subcolumn PD denotes the data of the $Primal-Dual$ algorithm reported in [20]. Note that exe lists the total running time of GFM_k. On the average, the GFM_k algorithm achieves 7% and 10% improvements on 3-way and 4-way partitioning, respectively, compared with the PD algorithm.

	3-way			4-way		
cir.	PD	GFM_k		PD	GFM_k	
	cut	cut	exe	cut	cut	exe
T2	81	81	158	217	182	183
T3	108	101	92	170	162	204
T4	100	104	127	154	138	147
T5	80	85	276	213	208	372
T6	157	104	101	189	145	196
PGA1	56	74	30	102	107	54
PGA2	377	259	222	459	335	396
PSC1	77	76	36	107	110	63
PSC2	370	261	211	426	354	443

Table 3: *Comparison with Primal-Dual approach.*

Comparison with HGCEP algorithm: The hierarchical gradual constraint-enforced partitioning algorithm ($HGCEP$) [18] is based on the two-phase approach. In the $HGCEP$ algorithm, the size constraints on the subsets are relaxed at the beginning and gradually enforced in later passes. In this experiment, the size of each cell was set to the real area size and the size overhead parameter was set to 20%. Table 4 lists the comparison results. The results of the GFM_{kt} algorithm are from one run only while the algorithm applied the GFM_k algorithm with the number of iterations 40 on clustered circuits. The GFM_{kt} algorithm achieves an average of 10% reduction.

	HGCEP	GFM_{kt}	
cir.	cut	cut	exe
T2	161	154	42
T3	132	117	34
T4	137	134	40
T5	201	144	51
T6	182	98	55
PGA1	96	100	21
PGA2	296	261	103
PSC1	89	100	24
PSC2	286	271	83

Table 4: *Comparison with HGCEP method.*

Comparison with net-based algorithm: The net-based algorithm [5] performs partitioning on a dual netlist representation. We used the test cases from the authors of [5]. The size overhead parameter was set to 10%. However, there are a few test cases where the area of the largest cell over the size upper bound. For these cases, the size overhead parameter needs to be relaxed. Since we were not able to get the exact size overhead parameter used in [5], we followed the suggestion in [11] that each large cell with size over the size upper bound will occupy a block and the rest of cells are partitioned into the remaining blocks under the size constraints (calculated from the total size of remaining cells and the number of remaining blocks). In this experiment, the number of iterations of the GFM_k

algorithm was set to 20. Table 5 shows the experimental results. We use DF to denote the net-based algorithm. Note that exe lists the total running time of GFM_k. On the average, GFM_k achieves 42% and 51% improvements over DF on 3-way and 4-way partitioning, respectively. In Table 5, * denotes those test cases with large cell violating the size constraints and the improvement over those test cases may not be exact.

	3-way			4-way		
	DF	GFM_k		DF	GFM_k	
cir.	cut	cut	exe	cut	cut	exe
T2	81	81	23	318	*179	136
T3	201	120	60	298	174	92
T4	329	*144	61	514	*223	97
T5	517	*166	118	863	*278	204
T6	252	156	66	285	144	101
PGA1	159	97	15	187	105	21
PGA2	578	268	114	717	422	143

Table 5: *Comparison with net based method DF on flat circuits; * denotes those test cases with large cell violating the size constraints.*

4.2 Partitioning with Replication

We performed the experiments on the same test cases as those in [12] where each I/O pad was set to the largest area of all cells. We set the size overhead limit to be 50%. In each table, *over* denotes the percentage of the cell size overhead due to replication.

We made comparison with the *FM on Replication Graph approach* ($FMRG$) [12]. The FMRG algorithm first construct a replication graph. Then, the directed FM algorithm is applied on the replication graph to generate a solution. Table 6 shows the experimental results. The number of the iterations in the GFM_r algorithm was set to 20. The results in the column GFM_{rt} are from one run only. On the average, the GFM_r and GFM_{rt} algorithms achieve 14% and 23% improvements over the $FMRG$ algorithm.

	FMRG	GFM_r			GFM_{rt}		
cir.	cut	cut	exe	$over$	cut	exe	$over$
T2	33	32	30	47%	31	19	48%
T3	23	17	26	49%	16	19	48%
T4	25	27	35	40%	23	18	38%
T5	34	23	72	34%	26	35	42%
T6	17	21	38	49%	11	23	47%
T7	29	18	76	48%	19	41	48%

Table 6: *Comparison with FMRG method.*

5 Acknowledgements

The authors would like to thank Professor J. Cong, Mr. W. Labio, N. Shivakumar, and B. M. Riess to provide the test cases. We also thank Dr. C. Yeh and C. Kring for their suggestions. The project is partially supported by NSF MIP-9315794 and Micro program.

References

[1] C. J. Alpert and S. Z. Yao, "Spectral Partitioning: The More Eigenvectors, The Better," *Proc.* 32^{th} *DAC*, 1995, pp. 195–200.

[2] T. Bui, C. Heigham, C. Jones, and T. Leighton, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms," *Proc.* 26^{th} *DAC*, 1989, pp. 775–778.

[3] R.E. Burkard and T. Bonniger, "A Heuristic for Quadratic Boolean Programs with Applications to Quadratic Assignment Problems," *European Journal of Operational Research*, 1983, 13, pp. 372–386.

[4] C.K. Cheng and Y.C. Wei, "An Improved Two-Way Partitioning Algorithm with Stable Performance," *IEEE Trans. on Computer-Aided Design*, 1991, pp. 1502–1511.

[5] J. Cong, W. Labio and N. Shivakumar, "Multi-Way VLSI Circuit Partitioning based on Dual Net Representation," *Proc. ICCAD*, 1994, pp. 56–62.

[6] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," *Proc.* 19^{th} *DAC*, 1982, pp. 175–181.

[7] L. Hargen and A. B. Kahng, "New Spectral Method for Ratio Cut Partitioning and Clustering," *IEEE Trans. on Compute-Aided Design*, 1991, pp. 1074–1085.

[8] J. Hwang and A. E. Gamal, "Optimal Replication for Min-Cut Partitioning", *Proc. ICCAD*, 1992, pp. 432–435.

[9] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, 1970, pp. 291–307.

[10] C. Kring and A. R. Newton, "A Cell-Replicating Approach to Mincut Based Circuit Partitioning," *Proc. ICCAD*, 1991, pp 2–5.

[11] W. Labio and N. Shivakumar, *Personal Communication*, Dec. 1994.

[12] L.T. Liu, M. T. Kuo, C.K. Cheng, and T. C. Hu, "A Replication Cut for Two-Way Partitioning," *IEEE Trans. on Computer-Aided Design*, 1995, pp. 623–632.

[13] S. Martello and P. Toth, *Knapsack Problems*, Chapter 7, Wiley, New York, 1990.

[14] B. M. Riess, K. Doll, and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques," *Proc.* 31^{th} *DAC*, 1994, pp. 646–651.

[15] L. A. Sanchis, "Multiple-Way Network Partitioning," *IEEE Trans. on Computers*, 1989, pp. 62–81.

[16] J. F. Shapiro, *Mathematical Programming: Structures and Algorithms*, Wiley, New York, 1979.

[17] M. Shih and E. S. Kuh "Quadratic Boolean Programming for Performance-Driven System Partitioning," *Proc.* 30^{th} *DAC*, 1993, pp. 761–765.

[18] H. Shin and C. Kim, "A Simple yet Effective Technique for Partitioning," *IEEE Trans. on Very Large Scale Integration System*, 1993 pp. 380–386.

[19] H. Yang and D. F. Wong, "Efficient Network Flow Based Min-Cut Balanced Partitioning," *Proc. ICCAD*, 1994, pp. 428–431.

[20] C.W. Yeh, C.K. Cheng, and T.T. Lin, "A General Purpose Multiple-Way Partitioning Algorithm," *IEEE Trans. on Computer-Aided Design*, 1994, pp. 1480–1488.