A Vertex Separator-based Algorithm for Hypergraph Bipartitioning

Enli Zhang
School of Computer Science and Technology, Xidian University, Xi'an, China
Email:yuleeo@163.com

Lin Gao
School of Computer Science and Technology, Xidian University, Xi'an, China
Email: lgao@mail.xidian.edu.cn

***Abstract*—Hypergraph partitioning is critical for dividing and conquering intractable problems in many complex systems, which is an NP-hard problem. In the paper, a novel hypergraph bipartitioning algorithm is proposed, which partitions the hypergraph by separating the intersection graph. The new approach completely eliminates the adverse effects of hyperedges with large cardinality on the performance of the vertex-oriented refinement algorithms, and enhances the hill-climbing ability of the move-based refinement algorithms. Our approach also simplifies the vertex designation. Experiments on the industrial testing benchmark datasets indicate that our approach achieves an average of 7% improvement in performance over FM family partitioning algorithms.**

***Index Terms*—Hypergraph Bipartitioning, Multileve Algorithm, Vertex Separator**

I. INTRODUCTION

Hypergraph is a type of generalized graph and its appearance fulfilled the demand of representing various complex informations in the mathematical and computer science problems, including Boolean Satisfiability, Numerical Linear Algebra, Machine Learning, Data Mining etc. Hypergraph partitioning is an important technology for dividing and conquering intractable problems in many complex systems. It is an NP-hard problem[2]. However, being promoted by some practical applications over the past half century, especially the electronic design automation (EDA), this problem has been attracting many researchers' attention, and a large number of heuristic algorithms have been developed. According to the various strategies used, the heuristic algorithms available can be roughly categorized into two classes: move-based approaches and mathematic approaches. The surveys by Alpert and Khang [3] and by David A. Papa and Igor L. Markov[4] provided detailed descriptions and comparisons of such various schemes.

The move-based approaches have been used widely. With an initial solution, these algorithms explore the solution space by locally perturbing the current solution and greedily exchange vertices that are most likely to improve the quality of the current solution. The earliest move-based algorithm was introduced to partitioning a graph by Kernighan and Lin (Abbr. KL Algorithm)[5] in 1970, and then this algorithm was adapted for solving the problem of circuit partitioning by other members of their group. Although the KL algorithm can produce good solutions, it is time-consuming. Fiduccia and Mattheyses (Abbr. FM Algorithm)[6] proposed a fast partitioning algorithm, which has improved the storage structure of the gain data and reduced the number of exchanged vertices, thus greatly improving its efficiency and allowing it can work nearly in linear time. Because of its simple implementation and low complexity of time, FM algorithm has become a widely used refinement tool in other move-based partitioning algorithms. Almost all algorithms of this class work directly in the hypergraph model, where hyperedges link typically to more than two vertices, and therefore the movement of one or more vertices need not instantly reduce the cut value, thus increasing the chance of strapped into local minima. As a result, it is difficult for the move-based refinement algorithms to remove the cut hyperedges with large cardinality, which straddle the partition with many vertices on both sides. Many researchers concentrate on the removal of the hyperedges that straddle the partition and have developed a lot of enhanced refinement algorithms, such as LA[7], CLIP/CDIP[8], LSR[9], PROP[10] , and so on. As they observed, it is more effective to move hyperedges than to move vertices, which leads to another approach to solve the problem of hypergraph partitioning, and our approach also concentrate on partitioning hyperedges directly. Furthermore, combining with stochastic optimizing strategies, many move-based hybrid partitioning algorithms have been proposed, including Simulated Annealing[11], Evolutionary Algorithms[12-15], Ant Colony Algorithm[16], Tabu Search[17]and the multi-start strategy. Although these stochastic algorithms have ability of global searching and can produce global optimum solution if given enough time, they take too long time to converge and hardly can be applied in

© 2014 ACADEMY PUBLISHER
doi:10.4304/jcp.9.8.1886-1896

practice.

The mathematic approaches directly construct the partitions by using mathematic methods, such as graph spectral algorithms [18,19,32] , mathematic programming, and network flowing (maxflow-mincut)[20]. As an important branch, spectral algorithms have been widely studied and applied in industrial fields. According to the graph spectral theory, the second smallest eigenvalue of the Laplace Matrix establishes a stable relationship with the optimal ratio cut partition and the corresponding eigenvector is referred as indicating vector. Generally, the spectral algorithms can produce global optimal solutions with natural partitioning border, but it is more suitable for ratio-cut metric. However, the construction of Laplace Matrix and the computing of indicating vector are two tricky things for spectral algorithms. One classic method is to convert hypergraph into weighted graph by replacing hyperedges with cliques and assigning each link a weight of $1/(|e|-1)$,where e is the cardinality of hyperedge [18]. However, this conversion only produces an approximately graph representation of the hypergraph, and it significantly increases the density of the resulting weighted graph. The computing of the main eigenvector is too expensive on time and space, which greatly hurts its applicability, even using fast Lanczos algorithm.

With the development of technology and the explosive growth of information, a latest technique for partitioning large scale hypergraphs, multilevel partitioning[21,22,23], was developed, which combines clustering algorithm with traditional move-based partitioning algorithm. The early multilevel partitioning algorithm only consists of two phases, namely, coarsening phase and partitioning phase. It uses clustering algorithm to coarse hypergraph by contracting groups of vertices into a series of single vertex, and then produces a coarsen hypergraph whose size is greatly reduced. During the partitioning phase, it directly partitions the coarsen hypergraph. Karypis and Kumar[21] adapted the early multilevel paradigm and proposed a three-phase algorithm, to which he introduced an uncoarsening and refinement process. Consequently, it is required to construct a sequence of successively coarser graphs during coarsening phase, after which a partition of the coarsest graph is produced. As the partition of next level inherits from the upper coarser hypergraph, the FM refinement algorithm is used to further refine the current partition at each level, which makes the multilevel paradigm even more robust. Furthermore, in their paper a hyperedge-oriented clustering method was proposed for speeding up the contraction of large hyperedges. Based on such new schemes, they developed the best known algorithm, hMetis. Since clustering can greatly reduce the size of hypergraphs, the performance of the hypergraph partitioning algorithms based on multilevel paradigm has been substantially improved. The multilevel algorithms are often two orders of magnitude faster than hitherto state-of-the-art partitioning algorithms. However, the performance of multilevel hypergraph partitioning algorithms depends heavily upon the quality of the clustering.

During the past decades, there appeared a minority who proposed a class of partitioning algorithms that partition the hypergraph by partitioning the hyperedges. The intersection graph of hyperedges (Abbr. IG graph) is used for the graph representation of a hypergraph, and the partition of hyperedges is directly computed in the IG graph. So far, there are only several literatures casting little effort on this approach. Kahng [24] provided a breath-first-searching method for searching the diameter of the IG graph and then yielding an approximate hyperedge partition. Hagen[25] used classic spectral partitioning algorithm directly to partition the IG graph. Cong [26] proposed a new way to construct the adjacent matrix of the IG graph and used the IG-matched method for determining the winner-loser of hyperedges. Recently, Cong[27] tried to implement a multi-way hypergraph partitioning by using a k-FM algorithm to partition the hyperedges in a hybrid hypergraph, which combines the IG graph with the dual hypergraph representation together. In his paper, a max-flow based method is adopted to solve the K-MC (K-Way Module Contention) problem. The hyperedge-oriented partitioning algorithm presents another efficient approach to hypergraph partitioning, as the IG graph is a precise graph representation of hypergraph.

In this paper, we propose a novel multilevel heuristic partitioning algorithm for hypergraph partitioning, and show that the vertex separator problem in the intersection graph representation is the dual problem of hypergraph partitioning. The new approach is a net-based multilevel partitioning algorithm using the IG graph representation of hypergraph. A move-based vertex separating algorithm is implemented to partition the IG graph for the hyperedges partition, and a community based coarsening scheme is introduced into the multilevel vertex separation. The proposed algorithm is experimentally shown to be readily applicable within the framework of intersection graph and computationally feasible.

The rest of this paper is organized as follows. Section 2 presents a formal description of the hypergraph bipartitioning problem and vertex separator. The necessary background of vertex separation also is given. Section 3 describes the proposed algorithm in detail. In section 4, a comparative qualitative analysis is made with experimental results given. Finally, Section 5 provides some concluding remarks and further research suggestions.

II. PRELIMINARIES

A. Hypergraph Bipartitioning

A hypergraph is a subsystem of the finite sets in combinatorics [1] and its conception derives from that of graph. A hypergraph $H=(V,E)$ is defined as a set of vertices V and a set of hyperedges E. Each hyperedge of the hypergraph is a subset of vertices and its cardinality is not limited in 2. Formally, a graph is a simple hypergraph whose edges own cardinality of 2. The partitioning of hypergraph is very similar to graph partitioning, and a large number of algorithms treat hypergraph partitioning

© 2014 ACADEMY PUBLISHER

as graph partitioning. For various applications, several metrics for hypergraph partitioning have been developed, and three basic partitioning formulations [3] are described as follows:

Minimum Width Bisection: Given a hypergraph *H*, the bisection is to partition its vertices *V* into two disjoint subsets A and B with $||A|-|B||\leq 1$, denoted as $P^2=\{A,B\}$, such that the number of hyperedges crossed the two partitions is minimized, denoted as $E(P^2)=\{e \mid e\in E, e\bigcap A\neq\varnothing, e\bigcap B\neq\varnothing\}$.

Minimum Balanced Cut: Given a hypergraph *H*, the bipartition is to separate its vertices *V* into two disjoint subsets A and B, such that $E(P^2)$ is minimized, subject to the balance constrain

$$L\leq\sum_{v\in A}w(v)\leq U\ ,\ L\leq\sum_{v\in B}w(v)\leq U$$

where $w(v)$ is the weight of vertex v, L and U denoted as the lower bound and upper bound respectively. Given a balance tolerance factor r,

$$L=(1-r)W/2,\ \ U=(1+r)W/2,$$

where W is the total vertex weight of the hypergraph.

Minimum Ratio-Cut: Given a hypergraph *H*, the bipartition is to separate its vertices *V* into two disjoint subsets A and B, such that $\dfrac{E(P^2)}{W(A)\cdot W(B)}$ is minimized, where $W(A)$ and $W(B)$ are the total vertex weights of the two partitions respectively.

Generally, one reasonable metric for hypergraph partitioning always count the cost between the cut value and the size of the disconnected partitions, and it often depends on specific applications. The above three metrics have different requirements for balance criteria. Bisection is the strictest partitioning, which constrains the deviation of two parts within one unit. This rigorous requirement is often unnecessary in practice and only leads to more complicated implements. On the contrary, the ratio cut metric greatly loosens requirements on balance criteria. Intuitively, the ratio cut metric allows partitioning algorithms to find optimum partition freely because the objective function makes a tradeoff between cut value and parts' size, which gives a soft penalty for uneven partition while capturing the minimum-cut. Since the size of the cut is on the numerator, it has more intensive effect on the objective value than the penalty does. Thus the ratio cut metric is apt to extremely unbalanced partition with smallest cut. While the balanced cut metric has a moderate balance criteria, which limits a legal area range by giving a balance tolerance factor, if the deviation of a partition is within the tolerance, it is regarded as a legal and acceptable solution. Practically, the balanced-cut metric is more flexible to meet different requirements of various applications by adjusting the balance factor. Therefore, it has been widely adopted by most hierarchal partitioning algorithms. Our approach also focuses on it.

B. Intersection Graph and Vertex Separator

Firstly, we consider the graph representation of a hypergraph. Practically, there are two types of graph that are widely used. One is a common weighted graph which is converted from the original hypergraph by replacing each hyperedge with a weighted clique, as [18] does. It is mostly used to construct the adjacent matrix in spectral partitioning algorithms, where each edge is assigned a connection weight. Actually, the weighted graph is a vertex intersection graph of the hypergraph, not an equivalent representation for hypergraph on the problem of partitioning, as mentioned above. Another is the hyperedge intersection graph which contains only the original hyperedges and their intersection information (Normally, we always refer to the hyperedge intersection graph as IG graph). It can be shown that the problem of hypergraph partitioning is really equivalent to the vertex separator problem in the IG graph, and they are a dual problem. The conversion from hypergraph to IG graph is as follows:

Given a hypergraph $H=(V,E)$ with $|V|=m$ and $|E|=n$, consider a graph with n vertices, denoted as $G=(V',E')$, such that the vertices V' of G represents the hyperedges E of H, one for each hyperedge, and the edges E' of G represents the intersection information between the corresponding hyperedges of H, i.e., if there is an edge between two vertices of G, in the original hypergraph H the intersection of the two corresponding hyperedges must not be empty, as Figure 2 demonstrates. G is defined as the hyperedge intersection graph of the hypergraph H. Note that this conversion is not bidirectional, and for a given hypergraph H, G is unique; however, the reverse is not true.

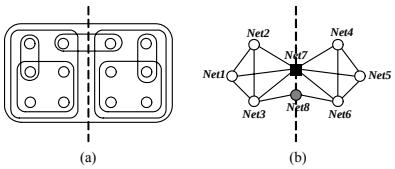

Figure 1. A hypergraph and its hyperedge intersection graph

As mentioned earlier, the objective of the hypergraph partitioning is to find a minimal cut, where the vertices are divided into two disjoint balanced parts. From another perspective, excluding the cut hyperedges that span two parts, the rest of the hyperedges also are divided into two disconnected parts in a partition. Therefore, a hypergraph partition can be obtained by separating its hyperedges rather than directly partitioning its vertices.

Given a partition of the hypergraph, the vertices are divided into two disconnected parts M and N, where the hyperedges are divided into three groups accordingly, with C as the cut hyperedges that straddle the partition, A and B as the hyperedges that are completely contained within the two disconnected parts M and N respectively. In the IG graph, the vertices corresponding to the group of hyperedges C form a separator, whose removal divides the rest vertices into two disconnected components corresponding to A and B respectively. On the contrary, suppose C is a separator of the IG graph, then the vertices are divided into two disconnected parts A and B by C. Projecting the partition about A, B and C to the original

© 2014 ACADEMY PUBLISHER

hypergraph, the hyperedges are divided into three corresponding parts. By assigning vertices that contained in the corresponding hyperedges of A and B to two vertices sets M and N respectively, the vertices of the hypergraph are divided into two parts, excluding few neutral unassigned vertices contained only in the hyperedges corresponding to separator C. Hence, it is easy to produce a partition for a separator of the IG graph. Therefore, considering hypergraph partitioning in terms of hyperedges, a separator of the IG graph corresponds uniquely to a hypergraph partition and the cut value of the hypergraph partition is completely consistent with the size of the separator. In conclusion, the problem of hypergraph partitioning is equivalent to the vertex separator problem in the IG graph, which means that they are a dual problem.

In balanced partitioning, the partition of the IG graph should satisfy such a balance constraint as guarantees the total weights of the vertices contained in the two corresponding hyperedges parts being equivalent. In this paper, we construct a weight IG graph by giving each vertex in the IG graph a capacity which approximately represents its weight, so that a balanced partition just requires that the total capacity of A and B approximately be equal. The balance constraint and vertex assignment are discussed in the next section. Since the vertex separator problem (VSP) is another important problem of graph partitioning and is also NP-Hard[28], it is intractable to find the minimal size vertex separators even in graphs with nodes having a maximum degree of three, to which we propose a novel move-based heuristic approach.

III. Multilevel Hypergraph Bipartitioning

Our approach VSHPA(Vertex Separator based Hypergraph Partitioning Algorithm) is based on multilevel separator model and applies the intersection graph representation of the hypergraph. As Figure 2 shows, the whole framework of VSHPA has five phases: IG graph construction, coarsening, initial separator, uncoarsening and refinement and partition projection.

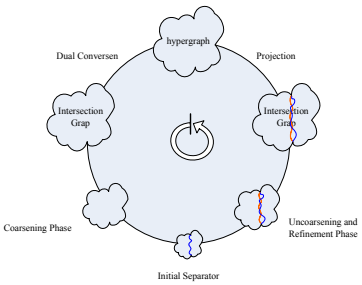

Figure 2. The various phases of VSHPA

At the preceding stage, the IG graph of the hypergraph needs to be constructed. One simple way is to replace hyperedges linked to one vertex with a clique in the IG graph. It takes $O(md^2)$ time to complete the whole conversion, where m is the number of vertices and d is the average degree of the vertices. Another fast conversion method is to search each hyperedge and its vertices one time, and add edge only to its adjacent hyperedges whose number is greater than itself. This method can avoid adding repeated edges and consumes time by $O(nd^2)$, where n is the number of hyperedges.

The intermediate three phases compose a multilevel vertex separating algorithm, and thus the whole procedure operate directly on the hyperedges in terms of graph vertices. In the coarsening phase, a community-based clustering method is used to coarsen the IG graph. Then an initial separator is produced by a fast agglomerating algorithm. A move-based heuristic algorithm is proposed specifically for refining the vertex separator in the uncoarsening and refinement phase.

At the last partition projection stage, a solution of the hypergraph partitioning is produced by assigning the vertices of hypergraph into partitions according to the hyperedge partitions projected from the minimal separator of the IG graph. In addition, we adopt an approximately statistic schemes to enforce the balance criteria in every cycle, including initial separator, refinement process and last vertex assignment. Several crucial methods are discussed in details in the following.

A. Community-based Coarsening Method

During the coarsening phase, groups of vertices are contracted into single vertices and a series of smaller coarsened graphs are produced, which greatly reduces the scale of the graph. However, the quality of clusters might greatly influence the final quality of the solution. Similar to the hypergraph representation, the IG graph preserves its hierarchical structure. Therefore, a community based clustering scheme is adopted for vertex separating, in which a group of vertices with maximal quality value are merged into an single hard core excluding boundary vertices adjacent to the rest of the graph, as Figure 3 shows.

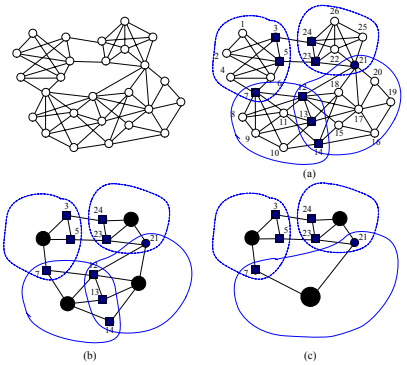

Figure 3. Illustration of the community based clustering scheme for vertex separation. The subsets of vertices circled by dashed lines are communities. The dark dots are the hard cores and the blue dots are the boundary vertices.

In the community based clustering scheme, the quality value of community is defined as $VQ = 1 - s / v^r$, where s is the number of the boundary vertices, v is the size of the community, and r is a control parameter for the size of the community denoted as $r = kd / \bar{d}$, where d is the density of the community, $\bar{d}$ is the average density of

© 2014 ACADEMY PUBLISHER

the graph, and k is a preference parameter for r. This definition aims to find a densely connected community with few boundary vertices, such that VQ reaches local maximal. Here parameter r can dynamically adjust the size of the community, which takes the density of the community into account, so denser community may have larger size. Getting such a community, we contract its internal vertices into a single hard core excluding the boundary vertices, which means that the vertices in the core are undividable and cannot be selected into separator, and that only the boundary vertices are potential members of the separator, even the last separator.

The clustering procedure has two stages. In the first stage, every vertex is visited and all the communities with maximal VQ are found. In the second stage, an agglomerative scheme is used to combine the communities to produce much coarser graph rather than to combine vertices. The communities with maximal VQ value are identified with the following procedures.

1) Randomly select an unvisited vertex, set it as the internal vertices and all its neighboring nodes as the boundary vertices;
2) Perform a loop over all boundary vertices to compute their fitness $f_i = VQ' - VQ$;
3) Add the boundary vertex with the largest fitness to the core with its unvisited neighboring nodes absorbed as new boundary vertices, yielding a larger community;
4) Repeat from step 2, until the quality value of the community reaches local maximal and there are no boundary vertices with positive fitness;
5) Cluster the core of the community into a hard core and mark all the vertices of the community as visited.

At the first stage, we repeat this procedure until all the vertices are visited. Such a coarsening progress turns the original intersection graph into a coarse graph consisting of many communities. The following stage continues to combine communities to produce a series of smaller coarse graphs until the smallest one reach the required size. During this process, pairs of communities with maximal fitness are selected to combine together if and only if the size of the new cluster meets the balance limit. To avoid big-bigger effect that bigger cluster is apt to absorb its adjacent cluster and grows up into supper large cluster, an upper limit is set to limit the size of the biggest cluster at every level.

B. Initial Separator

Generally speaking, the initial separator is not so important as it does not directly determine the last solution and might vary greatly during the uncoarsening and refinement process. It is a feasible way to execute the coarsening operation until only two communities are left, whose boundary vertices can be used as an initial separator, but it is hard to reach a balanced partition. Since the coarsest graph usually has a few vertices, it is easy for any separating algorithm to find a separator in seconds. In this paper, a fast agglomerating algorithm is used to compute the initial separator.

Firstly, two notations about the adjacent set of a vertex subset and the average out-degree of a separating vertex should be informed. The adjacent set of vertex subset X among vertex subset Y in a graph is defined as $N(X,Y) = \{v_j \mid v_i \in X, v_j \in Y, < v_i, v_j > \in E, i, j = 1..n\}$. Given a graph G, suppose C is a separator about A and B, for each separating vertex s in C, the average out-degree is defined as

$$Aod(s) = \sum_{v \in N(s,B)} \frac{1}{|N(v,C)|}.$$

The agglomerating algorithm starts with a randomly selected core vertex and sets the core as the internal vertex and all its boundary vertices as a separator. Each time, it selects a separating vertex with minimal average out-degree from the separator to combine with the internal vertices, and updates the separator. In moving a separating vertex, if its adjacent vertex is a contracted core, the core vertex should be uncut and must be moved immediately. The moving process continues until half of the vertices are moved into internal section or the total capacities of the internal vertices reach the upper balance limit. At that point, the vertices belonging to the internal section are assigned to the first partition, and the rest of the vertices are assigned to the second partition excluding the separator. The whole process of the algorithm can be summarized as follows:

1) Initialize an internal vertex set $S = \varnothing$ and a separator $C = \varnothing$;
2) Randomly select a core vertex as the seed, add it to S and its adjacent set $N(S, G - S)$ to C, and then produce a new separator about S and $G - S - C$;
3) Compute the average out-degree for all the separating vertices in C, select the vertex v_i with smallest value to move into S and add its adjacent set $N(v_i, G - S - C)$ to separator C;
4) Repeat step 3 until the size of S satisfies the requirement, and output the separator.

Since the initial separating algorithm is a randomized method, the quality of the separators is quite unstable and depends heavily on the seed vertex. In order to improve the quality of the partition, we start the next process with a small number of initial separators and drop some bad solutions during the uncoarsening and refinement phase. On one hand, the partition in the coarsest graph is just a proximate solution to lower level coarsen graph and will be substantially modified during the uncoarsening and refinement phase, and thus the optimal initial separator of the coarsest graph does not necessarily produce the smallest separator in the end. However, increasing the number of initial separator may potentially improve the quality of the final partitioning. On the other hand, a large number of initial separators will greatly increase the running time. Based on the observation that the size of coarser graph usually is very small and the refinement operation may be very fast, it is a wise decision to drop some bad solutions in the graph uncoarsening, which can effectively reduce the refinement time without substantial damage to the quality of the final solution.

© 2014 ACADEMY PUBLISHER

C. Refinement Algorithm for Vertex Separator

A move-based heuristic refinement algorithm VSR is implemented for vertex separator, which is a FM-liked stochastic refinement algorithm and directly optimizes the vertex separator, not the edge cut. Let C be a separator of graph G about two partitions R and L. Consider a subset Y of the separator C, move Y into L(here suppose vertices move from R to L), then the set $C'=(C-Y)\cup N(Y,R)$ is the new separator of the two partitions $R' = R - N(Y, R)$ and $L'=L\cup Y$. For $|N(Y,R)| < |Y|$, $|C'|<|C|$. It is the theoretic foundation of the refinement algorithms for vertex separator. J.W.H. Liu [29] provides first a Bipartite Graph Matching (BGM) algorithm in his paper. It uses the augment-path method to find a maximum matching between the Y and $N(Y, R)$. In finding the maximum matching, if there are uncovered vertices in Y and no augmenting paths, the vertex subset Y must satisfy the inequation $|N(Y,R)| < |Y|$. The BGM algorithm is a greedy searching algorithm and easily stuck into local minimal. As Figure 5 illustrates, the BGM algorithm can easily find the set C1 and P1. Upon replacing P1 with C1, the size of the separator is decreased by one. But it is difficult to find the set P2. Intuitively, successively moving C2 and P2 to L section,the subset C2 is replaced by P2′ and the size of the new separator $C'=(C-C_2)\cup N(P_2,R-P_2)$ is also reduced by one.

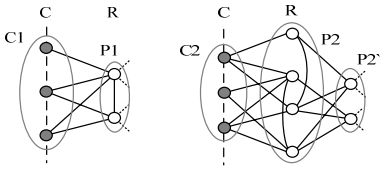

Figure 4. Illustration of the theory of refinement scheme for vertex separator

Based on such a fact, the theorem is extended to the general case: Suppose P is a vertex subset of R section, if $|N(P,C)| \geq |N(P,R-P)|$ and $N(N(P,C),R)\subseteq P$, move P to L section, and the separator is improved. Since the determination of such a vertex set P is intractable, a heuristic refinement approach is adopted for vertex separation. In our approach, a scoring scheme for each vertex is used to encourage closely adjacent groups of vertices to move together. This scheme is implemented as follows. Firstly, every vertex is given a relative score. Each time one vertex with maximal score is selected and moved, the scores of all the related vertices are updated by adding delta *s*. Suppose a vertex v is selected and moved into separator C, all the related vertices $N(N(v,C),R)$ need to be updated. A detailed illustration of the various steps of the refinement algorithm on a subgraph with 9 vertices is presented in Figure 5.

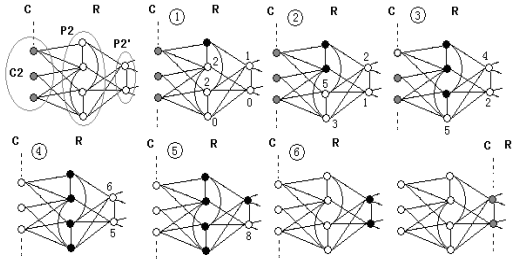

Figure 5. The refinement process of VSR in a simple subgraph with 9 vertices

At the beginning of the refinement process, the movements of selected vertices usually produce little reduction until a number of vertices are successively moved, which is known as the hill-climbing effect. Normally, the algorithm is likely to climb out of a local minimal and reaches a better solution after a number of vertices are moved. However, it is experimentally shown that too long a sequence of vertex moves rarely achieves an improvement of the separator. Usually, an adjustable parameter β is set to limit the number of such void moves and stop the refinement pass as soon as the void moves reach the upper limit. By adjusting the value of β in different refinement stage, the solution scope for refinement algorithm to search is dynamically changed. Since the initial solution may has lower quality at the beginning of the refinement process, a larger value for β can enlarge the searching scope for the refinement algorithm; whereas the solution derived from the upper level is finer in the lower level coarsen graph, a smaller value for β can limit the number of vertex moves, which can effectively speeds up refinement process.

Let C be a separator of graph G about two vertex sets R and L, the refinement algorithm for vertex separator works as follows:

1) Select the refining direction according to the size of the two partitions, and for $A_R > A_L$, select R side as the starting partition;
2) Initialize the adjacent vertices queue Q by $N(C,R)$ and set all the scores as zero;
3) Randomly select one vertex *v* from Q and move it to the separator C, add the unloaded neighbors in $N(v,R)$ to Q with zero score, and update the separator C and the queue Q;
4) Select the vertex *v* with maximal score to move into the separator C, add the unloaded neighbors in $N(v,R)$ to Q with zero score, and update the separator C and the queue Q;
5) Repeat step 4 until the total capacity of L section reach the upper limit or the void moves exceed the permitted number, put back all invalid moves and set them as inactive vertices to forbid them being selected as seed vertices again.
6) If there is no active vertex, current refinement process exit; else go to step 3 and continue.

D. Balance Constraint and Vertex Assignment

During the process of partitioning and refinement, the size of each partition must satisfy the balance constraint.

© 2014 ACADEMY PUBLISHER

In the IG graph, each vertex represents a hyperedge and one move means that a hyperedge is moved and a group of related vertices reassigned. Furthermore, there might be some neutral vertices that can be assigned to any partitions. It is hard to accurately compute the size of each partition. Therefore, it is feasible to approximately control the size of each partition during the IG graph separation. In this paper, a weighted IG graph is used to approximately compute the partition size by assigning each vertex a capacity A_i. For each vertex of the hypergraph, the average-degree-weight is defined as its average weight for each degree, and then the capacity of each hyperedge equals the total average-degree-weight of its vertices. Experimental data indicate that the weight of one vertex is proportional to its degree and the capacities of hyperedges can approximately represent its vertices' weights. It suffices to compute the capacity of each hyperedge once at the beginning of the partitioning, which greatly reduces the running time.

In the last process, the minimal separator of the IG graph is used to construct a hypergraph partition by assigning all the vertices of the hypergraph into partitions. A simple way is to assign the vertices according to the hyperedges' partition. One thing that has been left unspecified is how to assign the neutral vertices exclusively contained in the hyperedges of the separator. Because the assignations of the neutral vertices do not affect the cut value, they usually are assigned into the smaller partition to improve the balance condition.

IV. Experimental Results

The proposed algorithms are extensively tested on a large number of hypergraphs. The test data come mainly from the MCNC and ISPD98 standard benchmark suites [30, 31] and are widely used in ACM/SIGDA for testing hypergraph partitioning algorithms. Comparison between our approach and several known partitioning algorithms is performed with comprehensive results provided. Furthermore, supplementary experiments on community based clustering scheme are also made.

A. Hypergraph Test Benchmark

During the past decades, hypergraph partitioning has received enough attention from the Design Automation community and a large number of heuristic algorithms have been developed. Consequently, the requirement for appropriate testing data becomes stronger and hundreds of publications have used circuit benchmark suites to compare and validate their algorithms. There are a large number of circuits originally released by the Microelectronics Center of North Carolina (MCNC) and sponsored by ACM/SIGDA [30]. For convenient comparison of the quality of solutions, part of the MCNC and ISPD98 benchmark suites are selected to test our algorithm.

Table 4.1 presents the characteristics of 29 circuits of MCNC benchmark. The first 11 small scale circuits were introduced by MCNC before 1993 and are often used in the Physical Design Workshop. The latter 18 circuits of the ISPD98 benchmark suites [31] were introduced by Charles J. Alpert, member of IBM Austin Research Laboratory, in 1998. These selected circuits have a large range in scale, from 123 to 210,000 modules, and all are generated from real IBM internal designs. As the table shows, each circuit consists of cells, pads, pins and signals. Cells are the internal objects and occupy some circuit area, pads are the external objects of very small size (perhaps just external signal pins), and cell and pad are called the circuit module. Signals are the wires connecting circuit modules and transmitting electric signals, while pins are the feet points of circuit modules for connecting signals. The last column, Total Area, represents the size of circuits, i.e. the sum of all its modules' size. Using the hypergraph representation, each circuit can be naturally represented by a hypergraph, the vertices of the hypergraph representing the circuit modules and the hyperedges the signals. When using the real module's area, each vertex of the hypergraph is given a weight with the size of corresponding module.

TABLE 4.1

CHARACTERISTICS OF 29 CIRCUITS IN THE MCNC AND ISPD98 CIRCUIT TEST BENCHMARK

Benchmark	Cells	Pads	Signals	Pins	Total Area
ami33	33	-	123	388	1156449
ami49	49	-	408	912	35445424
fract	125	-	163	454	225504
g2	199	-	377	925	9596023
primary1	833	-	1266	3272	26607000
struct	1952	-	1920	5407	2850352
industry1	2271	-	2479	8025	4403352
primary2	3014	-	3817	12007	53457000
biomed	6514	-	5742	22253	13062992
industry2	12637	-	13419	47657	10404768
industry3	15406	-	21923	65610	360933376
ibm01	12752	246	14111	50566	4230016
ibm02	19601	259	19584	81199	8458336
ibm03	23136	283	27401	93573	9842880
ibm04	27507	287	31970	105859	9294944
ibm05	29347	1201	28446	126308	4471520
ibm06	32498	166	34826	128182	8577791
ibm07	45926	287	48117	175639	11829856
ibm08	51309	286	50513	204890	13449888
ibm09	53395	285	60902	222088	17529312
ibm10	69429	744	75196	297567	47534336
ibm11	70558	406	81454	280786	21237408
ibm12	71076	637	77240	317760	36974848
ibm13	84199	490	99666	357075	25061568
ibm14	147605	517	152772	546816	28727296
ibm15	161570	383	186608	715823	36534944
ibm16	183484	504	190048	778823	52592704
ibm17	185495	743	189581	860036	42187712
ibm18	210613	272	201920	819697	33686560

B. Clustering Results

During hypergraph partitioning, clustering is an important technology for reducing the size of the hypergraph, but how to judge the quality of the clusters has received little attention and there is no uniform criterion available. However, a versatile clustering scheme could effectively find tensely connected vertices and hyperedges subset and contract them into one vertex that should not be divided in most solutions. On the contrary, the hyperedges often cut in partitions should not be contracted, which are particularly helpful in searching the optimal solution by move-based refinement algorithms. Thus, it might be feasible to validate the

© 2014 ACADEMY PUBLISHER

clustering algorithms by analyzing and comparing the ratio of contracted hyperedges in a coarsen hypergraph. Our approach directly contracts communities in IG graph, which corresponds to directly contracting a group of tensely connected hyperedges together in the original hypergraph. For comparing the performance with other vertex-based clustering approach, Table 4.2 presents the ratios of contracted hyperedges when the vertices of a hypergraph are reduced to 80% by using heavy-edge matching(HEM), hMetis[21] and CMC respectively.

TABLE 4.2
COMPARISON OF HYPEREDGE CONTRACTED RATIOS AMONG HEM, HMETIS AND CMC.

Benchmark	Number of Hyperedge	Hyperedge Contracted Ratio (%)		
		HEM	hMetis	CMC
ibm01	14111	90.46	89.17	85.72
ibm02	19584	89.21	87.32	84.37
ibm03	27401	90.83	88.94	86.55
ibm04	31970	89.42	87.23	84.27
ibm05	28446	89.68	88.25	85.14
ibm06	34826	90.35	88.34	86.42
ibm07	48117	90.54	89.08	87.50
ibm08	50513	91.63	88.76	85.73
ibm09	60902	89.27	86.14	85.90
ibm10	75196	88.12	85.41	84.62
ibm11	81454	91.29	88.06	84.82
ibm12	77240	90.89	87.52	86.03
ibm13	99666	92.74	89.37	87.55
ibm14	152772	90.36	88.94	87.01
ibm15	186608	91.21	87.75	86.46
ibm16	190048	89.65	88.33	85.07
ibm17	189581	89.38	87.85	86.74
ibm18	201920	89.44	87.49	85.31

As Table 4.2 shows, our approach acquires the lowest overall hyperedge contraction ratio, indicating that it can effectively contract tensely connected vertices together while leaving more free hyperedges. We conjecture that the left hyperedges are the boundary vertices of communities in the IG graph, which are usually cut in local partitioning solutions.

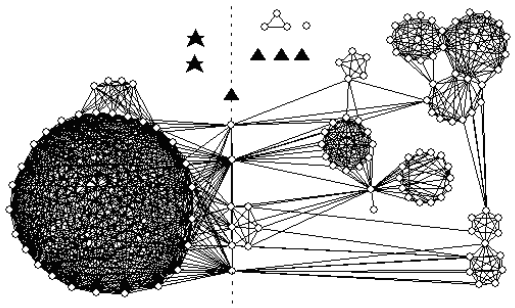

Figure 6. Communities and hierarchy in the intersection graph of circuit ami33.

In order to validate this conjecture, the IG graph of ami33 is extensively analyzed, as shown in Figure 6. There are 123 vertices in the IG graph. Specially, the two stars represent the ground and power signal, which link to all the other vertices of the graph; and the four triangles represent control signals that link to all vertices on the right of the dashed line. For a clear perspective, the links to the six vertices are hidden. Objectively, different from random systems, circuits are special manmade artifacts and they possess hierarchy structure and many highly structured communities, and therefore the community based clustering scheme can be applied in the IG graph.

C. Partitioning Comparison

To compare the quality of the solutions, we run three partitioners published earlier (Fiduccia-Mattheyses (FM) [6], IG-Match [26], hMetis [21]) and our VSHPA on the standard testing circuits. FM and hMetis are two vertex-oriented partitioners that work directly on hypergraph. FM is an industrial standard iterative exchange heuristic and its implementations use a LIFO bucket structure as [6] described. hMetis is a multilevel partitioner, whose executable program is obtained by the authors of [21] and whose adaptable parameters use the default schemes. Furthermore, for comparing the performance of different refinement schemes, we implement three different versions of VSHPA: $VSHPA^{BMP}$, $VSHPA^{Flat}$ and $VSHPA^{ML}$. $VSHPA^{BMP}$ is a common flat hypergraph partitioning algorithm adopting the BMP refinement method. $VSHPA^{Flat}$ is also a flat algorithm, but it uses the move-based refinement method to directly improve the separator. $VSHPA^{ML}$ is the multilevel version of $VSHPA^{Flat}$ and adopts the community based clustering method to coarsen the IG graph.

Firstly, we ran the six partitioners on the benchmarks under the unweighted hypergraph model (all circuit modules are regarded as standard cells with unit area). For each circuit, all the results allow up to 10% deviation from exact bisection, i.e., each partition must have an area between 45% and 55% of the total. Each partitioner ran 30 times and the best solutions are reported in Table 4.3. The column labeled "Hypergraph Model" shows that the contained partitioners work directly on hypergraphs, and the column labeled "Intersection Graph Model" indicates that the contained partitioners use the IG graphs of hypergraphs.

All the hyperedge-oriented algorithms perform better than the vertex-oriented algorithms. For flat partitioners, the quality of the solutions produced by $VSHPA^{BMP}$, $VSHPA^{Flat}$ and IG-match are better than that by FM algorithm with 6%, 17% and 3% improvement respectively, which further demonstrates that the cut hyperedges with large cardinality are are difficult to remove by the vertex-oriented refinement algorithms. For multilevel partitioners, the quality of the solutions produced by $VSHPA^{ML}$ is as good as that by hMetis, or even better, such as ibm03, ibm15, ibm17. Furthermore, hMetis perform better than FM algorithm, which hints that clustering scheme can effectively reduce the hyperedges' size and greatly improve the global searching ability of the vertex-oriented refinement algorithms. Generally speaking, our refinement algorithm performs better than FM and BMP based refinement algorithms. When the size of the test data is less than 1000, the three refinement algorithm all produce much better solutions,

© 2014 ACADEMY PUBLISHER

but the quality of the solutions produced by FM algorithm rapidly get worse with increasing size of the test data, which shows that the global searching ability of FM algorithm is worse on large scale data. Although the BMP based refinement algorithm also has better global searching ability, it is a greedily searching algorithm and easily trapped into local minimal. Our refinement algorithm has much better climbing ability than BMP.

TABLE 4.3
MIN-CUT BIPARTITIONING RESULTS UNDER THE UNWEIGHTED HYPERGRAPH MODEL WITH UP TO 10% DEVIATION FROM EXACT BISECTION. EACH VERTEX IS ASSIGNED THE UNIT AREA.

Benchmark	Hypergraph Model		Intersection Graph Model			
	FM	hMetis	IG-match	VSHPA^{BMP}	VSHPA^{Flat}	VSHPA^{ML}
ami33	10	9	9	10	10	9
ami49	47	46	46	49	49	46
fract	12	11	11	12	11	11
g2	29	16	24	36	29	18
primary1	68	47	53	72	60	41
struct	33	33	33	36	33	33
industry1	26	19	21	37	20	19
primary2	183	144	161	206	172	144
biomed	88	83	85	92	86	83
industry2	211	174	192	251	182	175
Industry3	273	264	283	242	187	260
ibm01	197	181	185	216	181	180
ibm02	266	262	294	293	264	262
ibm03	1151	956	1270	1481	1005	952
ibm04	602	537	611	1321	634	534
ibm05	1874	1739	1823	1935	1831	1723
ibm06	976	885	1061	1158	965	885
ibm07	1035	848	1084	1173	984	841
ibm08	1285	1142	1183	1276	1183	1148
ibm09	916	628	709	1082	709	626
ibm10	1502	1269	1518	1905	1517	1256
ibm11	1459	962	1296	1414	1209	962
ibm12	2258	1891	2273	2441	2219	1922
ibm13	1181	841	2126	2303	1025	840
ibm14	2961	1928	2469	3047	2490	1967
ibm15	5018	2748	4327	4423	3436	2597
ibm16	2363	1758	2383	2177	2383	2051
ibm17	3052	2341	3521	3839	3319	2206
ibm18	1708	1526	2128	2905	2549	1521

Table 4.4 presents bipartitioning results produced by partitioners under the weighted hypergraph model, where each vertex has actual module area. All the algorithms perform under the same circumstances as above. Note that the FM and IG-match algorithms sometimes produce worse solutions than that under unit area model, e.g., ibm05, ibm12 and ibm15, which indicates that the implementations are not particularly good at satisfying balance criteria when the areas of modules vary greatly. IG-match always forces all vertex moves to satisfy the balance criteria. Actually, it is more suitable for ratio cut metrics, under which it can find more natural partitions. Indeed, the problem of finding an exact bisection is NP-Complete under the weighted hypergraph model.

TABLE 4.4
MIN-CUT BIPARTITIONING RESULTS UNDER THE WEIGHTED HYPERGRAPH MODEL WITH UP TO 10% DEVIATION FROM EXACT BISECTION. EACH VERTEX IS ASSIGNED THE ACTUAL MODULE AREA.

Benchmark	Hypergraph Model		Intersection Graph Model			
	FM	hMetis	IG-match	VSHPA^{BMP}	VSHPA^{Flat}	VSHPA^{ML}
ami33	11	9	10	10	10	9
ami49	31	31	31	32	31	31
fract	12	11	11	12	11	11
g2	32	28	29	36	29	28
primary1	67	56	58	72	60	56
struct	34	33	33	36	33	33
industry1	23	20	20	37	20	20
primary2	198	161	202	206	172	160
biomed	88	83	88	92	86	83
industry2	252	168	211	351	182	166
Industry3	229	188	192	242	187	188
ibm01	272	216	251	216	181	216
ibm02	315	273	285	293	264	248
ibm03	1424	739	1009	1481	1005	694
ibm04	634	442	636	1321	634	440
ibm05	1878	1710	1922	1935	1831	1712
ibm06	1479	367	446	1158	965	363
ibm07	870	745	948	1573	984	716
ibm08	1411	1156	1249	1276	1183	1135
ibm09	750	522	708	2682	709	520
ibm10	982	734	1004	1905	1517	744
ibm11	1209	703	927	3414	1209	692
ibm12	2219	1988	2316	2441	2219	1975
ibm13	1196	874	911	2303	1025	850
ibm14	2015	1514	1930	5047	2490	1508
ibm15	3436	1802	3743	6423	3436	1781
ibm16	2173	1707	2039	4177	2383	1654
ibm17	2818	2247	3042	4839	3319	2253
ibm18	2604	1528	2511	2905	2549	1523

V. CONCLUSIONS AND FUTURE WORK

We present a new approach to hypergraph bipartitioning based on the combination of vertex separator and the IG graph representation of hypergraph. It is shown that the problem of hypergraph bipartitioning is the dual problem of vertex separation in the IG graph of hypergraph. Our vertex separator based hypergraph partitioning algorithm is hyperedge-oriented, which completely avoids the removal of large cut hyperedges. The results show that our approach performs better than the vertex-oriented partitioning algorithms. Furthermore, our approach greatly simplifies the vertex assignment process.

However, there are a number of interesting issues remain unsolved. According to the max-flow min-cut theorem, the problem of minimum ratio cut and vertex separator for some special graphs can be solved in time $O(n^{7/6}m^{2/3})$, but the problem of balanced vertex separator is NP-Hard. Although our heuristic approach also can produce high quality solutions, the randomized optimizing method enhances its global searching ability at the cost of its stability. It is feasible to improve its performance by increasing attempt times, but it brings more time complexity. Therefore, it is a promising research to find deterministic refinement algorithms for vertex separator. Since there are many special industrial application requirements, the problem of multi-way graph partitioning also deserves further study.

© 2014 ACADEMY PUBLISHER

ACKNOWLEDGEMENTS

This work was supported by the National Key Natural Science Foundation of China (Grant No. 60933009), Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 200807010013) and the National Natural Science Foundation of China (Grant No.60970065). We would like to thank our colleagues at our research labs for useful comments and assistance. Our gratitude also goes to Professors George Karypis and Vipin Kumar of University of Minnesota for supplying the hMetis executable.

REFERENCES

[1] C. BERGE, Graphs and Hypergruphs[M]. North-Holland, Amsterdam, 1973.
[2] M. R. Garey and D. S. Johnson.Computers and Intractability: A Guide to the Theory of NP-Completeness[M], San Francisco, CA: Freeman,1979.
[3] Charles J.Alpert and Andrew B.Kahang. Recent directions in netlist partitioning [J], Integration, the VLSI Journal, 1995,19(1-2),1-81.
[4] David A. Papa and Igor L. Markov, Hypergraph Partitioning and clustering.
[5] Kernighan B W, Lin S. An Efficient Heuristic Procedure for Partitioning Graphs [J]. Bell System Technical Journal. 1970, 49(2):291-307.
[6] C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions[C]. Proceedings of the 19th Conference on Design Automation, Piscataway, CA. 1982:175-181.
[7] B. Krishnamurthy. An improved mincut algorithm for partitioning VLSI networks [J], IEEE Trans. on Computer, 1984, 33 (5), 438-446.
[8] S. Dutt and W. Deng. VLSI circuit partitioning by cluster-removal using iterative improvement techniques [M]. In Proc. Physical Design Workshop,1996.
[9] 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[C]. IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, USA. 1997, 441-446.
[10] S. Dutt and W. Deng. Probability-based approach to VLSI circuit partitioning [J]. IEEE Trans. CAD, 2000, 19 (5): 534-549.
[11] D.S. Johnson, C.R. Aragon, L.A. McGeoch et al. Optimization by Simulated Annealing: An Experimental Evaluation, Part I, Graph Partitioning[J].Operations Research, 1989(37): 865-892.
[12] Y.Saab and V.Rao. Fast effective heuristics for the graph bisectioning problem [J]. IEEE Trans.Computer Aided Design, 1990, 9(1):91-98.
[13] T.N. Bui and B.R. Moon. Genetic Algorithm and Graph Partitioning [J].IEEE Trans. Computers, 1996(45): 841-855.
[14] Charles J Alpert, Lars W Hagen and Andrew B Kahng. A hybrid multilevel genetic approchoach for circuit partitioning[C]. Proceedings of IEEE Asia Pacific Conference on Circuit and Systems, Las Vegas, Nevada, USA. 1996:18~21.
[15] L.Tao, Y.C.Zhao, K. Thulasiraman, and M.N.S.Swamy. An efficient tabu search algorithm for graph bisectioning[C]. Proceedings of the First Great Lakes Symposium on VLSI, Kalamazoo, Michigan, 1991:92-95.
[16] Lin Gao,Yan Zeng and Anguo Dong. An ant colony algorithm for solving Max-cut problems [J]. Progress in Natural Science, 2008, 18 (9):1173-1178
[17] E. Rolland, H. Pirkul, and F. Glover.Tabu Search for Graph Partitioning [J]. Annals of Operational Research, 1996, 63(2): 209-232.
[18] T. Lengauer, Combinatorsal Algorithms for Integrated Circuit Layout, Wiley-Teubner, 1990.
[19] L. Hagen and A. B. Kahng. Fast spectral methods for ratio cut partitioning and clustering. In Proc. IEEE Intl. Conf. Computer-Aided Design, pages 10-13, 1991.
[20] J. Hwang and A. El Gamal. Optimal replication for min-cut partitioning. IEEE Trans. Computer-Aided Design, 14(1):96-106, 1995.
[21] George Karypis, Rajat Aggarwal and Vipin Kumar. Multilevel hypergraph partitioning: Application in VLSI domain [J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 1999, 3, 7(1). 69-79.
[22] Karypis George, Kumar Vipin. Multilevel k-way hypergraph partitioning [J]. VLSI Design. 2000, 11(3). 285-300.
[23] Charles J. Alpert, Jen-Hsin Huang, and Andrew B. Kahng. Multilevel Circuit Partitioning [J], IIEEE Trans.Computer Aided Design, 1998, 17(8):655-667.
[24] A. B. Kahng, "Fast hypergraph partition," in Proc. ACM/IEEE Design Automation Conf., 1989,pp. 762-764.
[25] L. Hagen and A. B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering", IEEE Trans. on CAD, pp.1074-1085, Sept. 1992.
[26] J.Cong,L.Hagen,and A.B.Kahng. Net partitions yield better module partitions[C].In Proc. IEEE 29th Design Automation Conf. Anaheim, California, USA,1992:47-52.
[27] J. CONG, LABIO W. J., and SHIVAKUMAR N. Multi-Way VLSI Circuit Partitioning Based on Dual Net Representation [J]. IEEE Trans. Computer-aided design of integrated circuits and systems, 1996, 15(4):396-409.
[28] Bui, T.N.; Jones, C. Finding good approximate vertex and edge partitions is NP-hard. Information Processing Letters 1992, 42, 153-159.
[29] Liu J W H. A graph partitioning algorithm by node separators [J]. ACM Trans on Mathematical Software, 1989, 15(3):198-219.
[30] http://www.ee.utulsa.edu/~tmanikas/MCNC/MCNC_Benchmark_Netlists.html.
[31] C. J. Alpert. The ISPD98 circuit benchmark suite[C]. In Proc. of the Intl Symposium of Physical Design, Monterey, CA USA, 1998(4): 80-85.
[32] Lliu X, Ma F, Lin H. Topic Detection with Hypergraph Partition Algorithm [J]. Journal of Software, 2011, 6(12): 2407-2415.

Enli Zhang received the B.S. degree in computer application from the Computer Science and Technology Department, Xi'an Technology Institute in 2001. He is currently working toward the Ph.D. degree in computer science at Xidian University.

His research interests include partitioning and clustering in VLSI circuit design, and the complex network structure.

© 2014 ACADEMY PUBLISHER

Lin Gao received the B.Sc. and M.Sc. in Computational Mathematics from Xi'an Jiaotong University and Northwest University in 1987 and 1990, respectively, and the Ph.D. degree in Circuit and System from School of Electronic Engineering, Xidian University in 2003.

She was a visiting scholar at University of Guelph, Canada from 2004 to 2005. Currently, she is an academic leader and professor in the School of Computer Science and Technology, Xidian University. Her research interests include bioinformatics, data mining in biological data, graph theory and intelligence computation.

© 2014 ACADEMY PUBLISHER