An Effective Multi-level Algorithm Based on Ant Colony Optimization for Bisecting Graph

Ming Leng and Songnian Yu

School of Computer Engineering and Science,
Shanghai University, Shanghai, PR China 200072
lengming@graduate.shu.edu.cn,
snyu@staff.shu.edu.cn

Abstract. An important application of graph partitioning is data clustering using a graph model — the pairwise similarities between all data objects form a weighted graph adjacency matrix that contains all necessary information for clustering. The min-cut bipartitioning problem is a fundamental graph partitioning problem and is NP-Complete. In this paper, we present an effective multi-level algorithm based on ant colony optimization(ACO) for bisecting graph. The success of our algorithm relies on exploiting both the ACO method and the concept of the graph core. Our experimental evaluations on 18 different graphs show that our algorithm produces encouraging solutions compared with those produced by MeTiS that is a state-of-the-art partitioner in the literature.

1 Introduction

An important application of graph partitioning is data clustering using a graph model [1], [2]. Given the attributes of the data points in a dataset and the similarity or affinity metric between any two points, the symmetric matrix containing similarities between all pairs of points forms a weighted adjacency matrix of an undirected graph. Thus the data clustering problem becomes a graph partitioning problem [2]. The *min-cut bipartitioning problem* is a fundamental partitioning problem and is NP-Complete [3]. It is also NP-Hard to find good approximate solutions for this problem [4]. Because of its importance, the problem has attracted a considerable amount of research interest and a variety of algorithms have been developed over the last thirty years [5],[6]. The survey by Alpert and Kahng [7] provides a detailed description and comparison of various such schemes which can be classified as *move-based* approaches, *geometric representations*, *combinatorial* formulations, and *clustering* approaches.

Most existing partitioning algorithms are heuristics in nature and they seek to obtain reasonably good solutions in a reasonable amount of time. Kernighan and Lin (KL) [5] proposed a heuristic algorithm for partitioning graphs. The KL algorithm is an iterative improvement algorithm that consists of making several improvement passes. It starts with an initial bipartitioning and tries to improve it by every pass. A pass consists of the identification of two subsets of vertices, one from each part such that can lead to an improved partition if

Z.-H. Zhou, H. Li, and Q. Yang (Eds.): PAKDD 2007, LNAI 4426, pp. 138–149, 2007.

the vertices in the two subsets switch sides. Fiduccia and Mattheyses (FM) [6] proposed a fast heuristic algorithm for bisecting a weighted graph by introducing the concept of cell *gain* into the KL algorithm. These algorithms belong to the class of *move-based* approaches in which the solution is built iteratively from an initial solution by applying a move or transformation to the current solution. Move-based approaches are the most frequently combined with stochastic hill-descending algorithms such as those based on Simulated Annealing [8], Tabu Search [8],[9], Genetic Algorithms [10], Neural Networks [11], etc., which allow movements towards solutions worse than the current one in order to escape from local minima. For example, Leng and Yu [12],[13] proposed a boundary Tabu Search refinement algorithm that combines an effective Tabu Search strategy with a boundary refinement policy for refining the initial partitioning.

As the problem sizes reach new levels of complexity recently, it is difficult to compute the partitioning directly in the original graph and a new class of graph partitioning algorithms have been developed that are based on the multi-level paradigm. The multi-level graph partitioning schemes consist of three phases [14],[15],[16]. During the *coarsening phase*, a sequence of successively coarser graph is constructed by collapsing vertex and edge until its size is smaller than a given threshold. The goal of the *initial partitioning phase* is to compute initial partitioning of the coarsest graph such that the balancing constraint is satisfied and the partitioning objective is optimized. During the *uncoarsening phase*, the partitioning of the coarser graph is successively projected back to the next level finer graph and an iterative refinement algorithm is used to optimize the objective function without violating the balancing constraint.

In this paper, we present a multi-level algorithm which integrates an effective matching-based coarsening scheme and a new ACO-based refinement approach. Our work is motivated by the multi-level *ant* colony algorithm(MACA) of Korošec who runs basic *ant* colony algorithm on every level graph in [17] and Karypis who introduces the concept of the graph *core* for coarsening the graph in [16] and supplies **MeTiS** [14], distributed as open source software package for partitioning unstructured graphs. We test our algorithm on 18 graphs that are converted from the hypergraphs of the ISPD98 benchmark suite [18]. Our comparative experiments show that our algorithm produces excellent partitions that are better than those produced by **MeTiS** in a reasonable time.

The rest of the paper is organized as follows. Section 2 provides some definitions and describes the notation that is used throughout the paper. Section 3 briefly describes the motivation behind our algorithm. Section 4 presents an effective multi-level ACO refinement algorithm. Section 5 experimentally evaluates our algorithm and compares it with **MeTiS**. Finally, Section 6 provides some concluding remarks and indicates the directions for further research.

2 Mathematical Description

A graph $G=(V,E)$ consists of a set of vertices V and a set of edges E such that each edge is a subset of two vertices in V. Throughout this paper, n and m denote

the number of vertices and edges respectively. The vertices are numbered from *1* to n and each vertex $v \in V$ has an integer weight $S(v)$. The edges are numbered from *1* to m and each edge $e \in E$ has an integer weight $W(e)$. A decomposition of a graph V into two disjoint subsets V^1 and V^2, such that $V^1 \cup V^2 = V$ and $V^1 \cap V^2 = \varnothing$, is called a *bipartitioning* of V. Let $S(A) = \sum_{v \in A} S(v)$ denotes the size of a subset $A \subseteq V$. Let ID_v be denoted as v's *internal degree* and is equal to the sum of the edge-weights of the adjacent vertices of v that are in the same side of the partition as v, and v's *external degree* denoted by ED_v is equal to the sum of edge-weights of the adjacent vertices of v that are in different sides. The *cut* of a *bipartitioning* $P = \{V^1, V^2\}$ is the sum of weights of edges which contain two vertices in V^1 and V^2 respectively. Naturally, vertex v belongs at the boundary if and only if $ED_v > 0$ and the *cut* of P is also equal to $0.5 \sum_{v \in V} ED_v$. Given a balance constraint r, the *min-cut bipartitioning problem* seeks a solution $P = \{V^1, V^2\}$ that minimizes *cut(P)* subject to $(1\text{-}r)S(V)/2 \leq S(V^1), S(V^2) \leq (1+r)S(V)/2$. A *bipartitioning* is *bisection* if r is as small as possible. The task of minimizing *cut(P)* can be considered as the *objective* and the requirement that solution P will be of the same size can be considered as the *constraint*.

3 Motivation

ACO is a novel population-based meta-heuristic framework for solving discrete optimization problems [19],[20]. It is based on the indirect communication among the individuals of a colony of agents, called *ants*, mediated by trails of a chemical substance, called *pheromone*, which real *ants* use for communication. It is inspired by the behavior of real *ant* colonies, in particular, by their foraging behavior and their communication through *pheromone* trails. The *pheromone* trails are a kind of distributed numeric information which is modified by the *ants* to reflect their experience accumulated while solving a particular problem. Typically, solution components which are part of better solutions or are used by many *ants* will receive a higher amount of *pheromone* and, hence, will more likely be used by the *ants* in future iterations of the algorithm. The collective behavior that emerges is a form of *autocatalytic* behavior. The process is thus characterized by a *positive feedback* loop, where the probability with which *ant* chooses a solution component increases with the number of *ants* that previously chose the same solution component.

The main idea of ACO is as follows. Each *ant* constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. At every point each *ant* has to decide which solution component to be added to its current partial solution according to a *state transition rule*. After the solution construction is completed, the *ants* give feedback on the solutions they have constructed by depositing *pheromone* on solution components which they have used in their solution according to a *pheromone updating rule*.

In [21], Langham and Grant proposed the Ant Foraging Strategy (AFS) for k-way partitioning. The basic idea of the AFS algorithm is very simple: We have k colonies of *ants* that are competing for food, which in this case represents the vertices of the graph. At the end the *ants* gather food to their nests, i.e. they partition the graph into k subgraphs. In [17], Korošec presents the MACA approach that is enhancement of the AFS algorithm with the multi-level paradigm. However, since Korošec simply runs the AFS algorithm on every level ℓ graph $G_l(V_l,E_l)$, most of computation on the coarser graphs is wasted. Furthermore, MACA comes into collision with the key idea behind the multi-level approach. The multi-level graph partitioning schemes needn't the direct partitioning algorithm on $G_l(V_l,E_l)$ in the *uncoarsening and refinement phase*, but the refinement algorithm that improves the quality of the finer graph $G_l(V_l,E_l)$ partitioning $P_{G_l}=\{V_l^1,V_l^2\}$ which is projected from the partitioning $P_{G_{l+1}}=\{V_{l+1}^1,V_{l+1}^2\}$ of the coarser graph $G_{l+1}(V_{l+1},E_{l+1})$.

In this paper, we present a new multi-level *ant* colony optimization refinement algorithm(MACOR) that combines the ACO method with a boundary refinement policy. It employs ACO in order to select two subsets of vertices $V_l^{1'} \subset V_l^1$ and $V_l^{2'} \subset V_l^2$ such that $\{ (V_l^1 - V_l^{1'}) \cup V_l^{2'} , (V_l^2 - V_l^{2'}) \cup V_l^{1'} \}$is a bisection with a smaller edge-cut. It has distinguishing features which are different from the MACA algorithm. First, MACA exploits two or more colonies of *ants* to compete for the vertices of the graph, while MACOR employs one colony of *ants* to find $V_l^{1'}$ and $V_l^{2'}$ such that moving them to the other side improves the quality of partitioning. Second, MACA is a partitioning algorithm while MACOR is a refinement algorithm. Finally, MACOR is a boundary refinement algorithm whose runtime is significantly smaller than that of a non-boundary refinement algorithm, since the vertices moved by MACOR are boundary vertices that straddle two sides of the partition and only the gains of boundary vertices are computed.

In [14], Karypis presents the sorted heavy-edge matching (SHEM) algorithm that identifies and collapses together groups of vertices that are highly connected. Firstly, SHEM sorts the vertices of the graph ascendingly based on the *degree* of the vertices. Next, the vertices are visited in this order and SHEM matches the vertex v with unmatched vertex u such that the weight of the edge $W(v,u)$ is maximum over all incident edges. In [22], Sediman introduces the concept of the graph *core* firstly that the *core* number of a vertex v is the maximum order of a *core* that contains that vertex. Vladimir gives an $O(m)$-time algorithm for cores decomposition of networks and $O(m\cdot \log(n))$-time algorithm to compute the *core* numbering in the context of sum-of-the-edge-weights in [23],[24] respectively. In [16], Amine and Karypis introduce the concept of the graph *core* for coarsening the *power-law* graphs. In [13], Leng present the core-sorted heavy-edge matching (CSHEM) algorithm that combines the concept of the graph *core* with the SHEM scheme. Firstly, CSHEM sorts the vertices of the graph descendingly based on the *core* number of the vertices by the algorithm in [24]. Next, the vertices are visited in this order and CSHEM matches the vertex v with its unmatched neighboring vertex whose edge-weight is maximum. In case of a tie according to edge-weights, we will prefer the vertex that has the highest *core* number.

In our multi-level algorithm, we adopt the MACOR algorithm during the *refinement phase* , the greedy graph growing partition (GGGP) algorithm [14] during the *initial partitioning phase*, an effective matching-based coarsening scheme during the *coarsening phase* that uses the CSHEM algorithm on the original graph and the SHEM algorithm on the coarser graphs. The pseudocode of our multi-level algorithm is shown in Algorithm 1.

Algorithm 1 (Our multi-level algorithm)

INPUT: original graph $G(V,E)$
OUTPUT: the partitioning P_G of graph G
/**coarsening phase*/*
$l = 0$
$G_l(V_l,E_l)$=$G(V,E)$
$G_{l+1}(V_{l+1},E_{l+1})$=CSHEM($G_l(V_l,E_l)$)
While ($|V_{l+1}| > 20$) do
  $l = l + 1$
  $G_{l+1}(V_{l+1},E_{l+1})$=SHEM($G_l(V_l,E_l)$)
End While
/**initial partitioning phase*/*
P_{G_l}=GGGP(G_l)
/**refinement phase*/*
While ($l \geq 1$) do
  $P^{'}_{G_l}$=MACOR(G_l,P_{G_l})
  Project $P^{'}_{G_l}$ to $P_{G_{l-1}}$;
  $l = l - 1$
End While
P_G=MACOR(G_l,P_{G_l})
Return P_G

4 An Effective Multi-level Ant Colony Optimization Refinement Algorithm

Informally, the MACOR algorithm works as follows: At time zero, an initialization phase takes place during which the internal and external degrees of all vertices are computed and initial values for *pheromone* trail are set on the vertices of graph G. In the main loop of MACOR, each *ant*'s tabu list is emptied and each *ant* chooses ($V^{1'}$, $V^{2'}$) by repeatedly selecting boundary vertices of each part according to a *state transition rule* given by Equation(1)(2), moving them into the other part, updating the gains of the remaining vertices and etc. After constructing its solution, each *ant* also modifies the amount of *pheromone* on the moved vertices by applying the *local updating rule* of Equation(3). Once all *ants* have terminated their solutions, the amount of *pheromone* on vertices is modified again by applying the *global updating rule* of Equation(4). The process is iterated until the cycles counter reaches the maximum number of cycles NC_{max}, or the MACOR algorithm stagnates.

The pseudocode of the MACOR algorithm is shown in Algorithm 2. The cycles counter is denoted by t and *Best* represents the best partitioning seen so far. The initial values for *pheromone* trail is denoted by τ_0=1/ε, where ε is total number of *ants*. At cycle t, let $\tau_v(t)$ be the *pheromone* trail on the vertex v and $tabu^k(t)$ be the tabu list of *ant* k, $Best^k(t)$ represents the best partitioning found by *ant* k and the current partitioning of *ant* k is denoted by $P^k(t)$, the *ant* k also stores the internal and external degrees of all vertices and boundary vertices independently which be denoted as $ID^k(t)$, $ED^k(t)$ and $boundary^k(t)$ respectively. Let $allowed^k(t)$ be denoted as the *candidate* list which is a list of preferred vertices to be moved by *ant* k at cycle t and is equal to $\{V - tabu^k(t)\} \bigcap boundary^k(t)$.

Algorithm 2 (MACOR)

INPUT: initial bipartitioning P, maximum number of cycles NC_{max}
balance constraint r, similarity tolerance φ, maximum steps s_{max}
OUTPUT: the best partitioning *Best*, cut of the best partitioning $cut(Best)$
/**Initialization**/
$t = 0$
$Best = P$
For every vertex v in $G = (V, E)$ do
$ID_v = \sum_{(v,u)\in E \wedge P[v]=P[u]} W(v,u)$
$ED_v = \sum_{(v,u)\in E \wedge P[v]\neq P[u]} W(v,u)$
Store v as *boundary vertex* if and only if $ED_v > 0$;
$\tau_v(t) = \tau_0$
End For
/**Main loop**/
For $t = 1$ to NC_{max} do
For $k = 1$ to ε do
$tabu^k(t) = \varnothing$
Store $P^k(t) = P$ and $Best^k(t) = P$ independently;
Store $ID^k(t)$, $ED^k(t)$, $boundary^k(t)$ of $G = (V, E)$ independently;
For $s = 1$ to s_{max} do
Decide the *move direction* of the current step s;
If exists at least one vertex $v \in allowed^k(t)$ then
Choose the vertex v to move as follows

$$v = \begin{cases} \arg \max_{v \in allowed^k(t)} [\tau_v(t)]^\alpha \cdot \left[\eta_v^k(t)\right]^\beta & \text{if } q \le q_0 \\ w & \text{if } q > q_0 \end{cases} \tag{1}$$

Where the vertex w is chosen according to the probablity

$$p_w^k(t) = \begin{cases} \frac{[\tau_w(t)]^\alpha \cdot \left[\eta_w^k(t)\right]^\beta}{\sum_{u \in allowed^k(t)} [\tau_u(t)]^\alpha \cdot [\eta_u^k(t)]^\beta} & \text{if } w \in allowed^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Else
Break;
End If
Update $P^k(t)$ by moving the vertex v to the other side;
Lock the vertex v by adding to $tabu^k(t)$;
original cut Minus its original *gain* as the *cut* of $P^k(t)$;
Update $ID_u^k(t)$, $ED_u^k(t)$, *gain* of its neighboring vertices u and $boundary^k(t)$;
If ($cut(P^k(t)) < cut(Best^k(t))$ and $P^k(t)$ satisfies constraints r) then $Best^k(t) = P^k(t)$
End If
End For /*$s \leq s_{max}$*/
Apply the *local update rule* for the vertices v moved by *ant* k

$$\tau_v(t) \leftarrow (1-\rho) \cdot \tau_v(t) + \rho \cdot \triangle\tau_v^k(t) \tag{3}$$

Adjust q_0 if $similarity((V^{1'}, V^{2'})^k, (V^{1'}, V^{2'})^{(k\text{-}1)}) \geq \varphi$;
End For /*$k \leq \varepsilon$*/
If $\min\limits_{1\leq k\leq \varepsilon} cut(Best^k(t)) < cut(Best)$ then
Update *Best* and $cut(Best)$;
End If
Apply the *global update rule* for the vertices v moved by *global-best ant*

$$\tau_v(t) \leftarrow (1-\xi) \cdot \tau_v(t) + \xi \cdot \triangle\tau_v^{gb} \tag{4}$$

For every vertex v in $G = (V, E)$ do
$\tau_v(t+1) = \tau_v(t)$
End For
End For /*$t \leq NC_{max}$*/
Return *Best* and $cut(Best)$

In the MACOR algorithm, a *state transition rule* given by Equation(1)(2) is called *pseudo-random-proportional rule*, where q is a random number uniformly distributed in $[0 \ldots 1]$ and q_0 is parameter $(0 \leq q_0 \leq 1)$ which determines the relative importance of exploitation versus exploration. If $q \leq q_0$ then the best vertex, according to Equation(1), is chosen(exploitation), otherwise a vertex is chosen according to Equation(2)(exploration). To avoid trapping into *stagnation behavior*, MACOR adjusts dynamically the parameter q_0 based on the solutions similarity between $(V^{1'}, V^{2'})^k$ and $(V^{1'}, V^{2'})^{(k\text{-}1)}$ found by *ant k* and *k-1*. In Equation(1)(2), α and β denote the relative importance of the *pheromone* trail $\tau_v(t)$ and *visibility* $\eta_v^k(t)$ respectively, $\eta_v^k(t)$ represents the *visibility* of *ant k* on the vertex v at cycle t and is given by:

$$\eta_v^k(t) = \begin{cases} \sqrt{1.0 + ED_v^k(t) - ID_v^k(t)} & \text{if } (ED_v^k(t) - ID_v^k(t)) \geq 0 \\ \sqrt{1.0/(ID_v^k(t) - ED_v^k(t))} & \text{otherwise} \end{cases} \tag{5}$$

In Equation(3), ρ is a coefficient and represents the local evaporation of *pheromone* trail between cycle t and $t+1$ and the term $\triangle\tau_v^k(t)$ is given by:

$$\triangle\tau_v^k(t) = \begin{cases} \frac{cut(Best^k(t)) - cut(P)}{cut(P)\cdot\varepsilon} & \text{if } v \text{ was moved by } ant\ k \text{ at cycle } t \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

In Equation(4), ξ is a parameter and represents the global evaporation of *pheromone* trail between *cycle* t and $t+1$ and the term $\triangle\tau_v^{gb}$ is given by:

$$\triangle\tau_v^{gb} = \begin{cases} \frac{cut(Best) - cut(P)}{cut(P)} & \text{if } v \text{ was moved by } global\text{-}best\ ant \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

5 Experimental Results

We use the 18 graphs in our experiments that are converted from the hypergraphs of the ISPD98 benchmark suite [18] and range from 12,752 to 210,613 vertices. Each hyperedge is a subset of two or more vertices in hypergraph. We convert hyperedges into edges by the rule that every subset of two vertices in hyperedge can be seemed as edge. We create the edge with unit weight if the edge that connects two vertices doesn't exist, else add unit weight to the weight of the edge. Next, we get the weights of vertices from the ISPD98 benchmark. Finally, we store 18 edge-weighted and vertex-weighted graphs in format of **MeTiS** [14]. The characteristics of these graphs are shown in Table 1.

Table 1. The characteristics of 18 graphs to evaluate our algorithm

benchmark	vertices	hyperedges	edges
ibm01	12752	14111	109183
ibm02	19601	19584	343409
ibm03	23136	27401	206069
ibm04	27507	31970	220423
ibm05	29347	28446	349676
ibm06	32498	34826	321308
ibm07	45926	48117	373328
ibm08	51309	50513	732550
ibm09	53395	60902	478777
ibm10	69429	75196	707969
ibm11	70558	81454	508442
ibm12	71076	77240	748371
ibm13	84199	99666	744500
ibm14	147605	152772	1125147
ibm15	161570	186608	1751474
ibm16	183484	190048	1923995
ibm17	185495	189581	2235716
ibm18	210613	201920	2221860

We implement the MACOR algorithm in ANSI C and integrate it with the leading edge partitioner **MeTiS**. In the evaluation of our multi-level algorithm, we must make sure that the results produced by our algorithm can be easily compared against those produced by **MeTiS**. We use the same balance constraint r and random seed in every comparison. In the scheme choices of three phases offered by **MeTiS**, we use the SHEM algorithm during the *coarsening phase*, the GGGP algorithm during the *initial partitioning phase* that consistently finds smaller edge-cuts than other algorithms, the boundary KL (BKL) refinement algorithm during the *uncoarsening and refinement phase* because BKL can produce smaller edge-cuts when coupled with the SHEM algorithm. These measures are sufficient to guarantee that our experimental evaluations are not biased in any way.

Table 2. Min-cut bipartitioning results with up to 2% deviation from exact bisection

benchmark	MeTiS(α)		our algorithm(β)		ratio(β:α)		improvement	
	MinCut	AveCut	MinCut	AveCut	MinCut	AveCut	MinCut	AveCut
ibm01	517	1091	259	531	0.501	0.487	49.9%	51.3%
ibm02	4268	11076	1920	5026	0.450	0.454	55.0%	54.6%
ibm03	10190	12353	4533	5729	0.445	0.464	55.5%	53.6%
ibm04	2273	5716	2221	3037	0.977	0.531	2.3%	46.9%
ibm05	12093	15058	8106	9733	0.670	0.646	33.0%	35.4%
ibm06	7408	13586	2111	5719	**0.285**	0.421	**71.5%**	57.9%
ibm07	3219	4140	2468	3110	0.767	0.751	23.3%	24.9%
ibm08	11980	38180	10500	13807	0.876	**0.362**	12.4%	**63.8%**
ibm09	2888	4772	2858	3905	0.990	0.818	1.0%	18.2%
ibm10	10066	17747	5569	7940	0.553	0.447	44.7%	55.3%
ibm11	2452	5095	2405	3423	0.981	0.672	1.9%	32.8%
ibm12	12911	27691	5502	13125	0.426	0.474	57.4%	52.6%
ibm13	6395	13469	4203	6929	0.657	0.514	34.3%	48.6%
ibm14	8142	12903	8435	10114	**1.036**	0.784	**-3.6%**	21.6%
ibm15	22525	46187	17112	25102	0.760	0.543	24.0%	45.7%
ibm16	11534	22156	8590	12577	0.745	0.568	25.5%	43.2%
ibm17	16146	26202	13852	18633	0.858	0.711	14.2%	28.9%
ibm18	15470	20018	15494	18963	1.002	**0.947**	-0.2%	**5.3%**
average					**0.721**	**0.589**	**27.9%**	**41.1%**

The quality of partitions produced by our algorithm and those produced by **MeTiS** are evaluated by looking at two different quality measures, which are the minimum *cut* (MinCut) and the average *cut* (AveCut). To ensure the statistical significance of our experimental results, two measures are obtained in twenty runs whose random seed is different to each other. For all experiments, we allow the balance constraint up to 2% deviation from exact bisection by setting r to 0.02, i.e., each partition must have between 49% and 51% of the total vertices size. We also set the number of vertices of the current level graph as the

value of parameter s_{max}. Furthermore, we adopt the experimentally determined optimal set of parameters values for MACOR, α=2.0, β=1.0, ρ=0.1, ξ=0.1, q_0=0.9, φ=0.9, NC_{max}=80, ε=10.

Table 2 presents *min-cut bipartitioning* results allowing up to 2% deviation from exact bisection and Fig. 1 illustrates the MinCut and AveCut comparisons of two algorithms on 18 graphs. As expected, our algorithm reduces the AveCut by 5.3% to 63.8% and reaches 41.1% average AveCut improvement. Although our algorithm produces partition whose MinCut is up to 3.6% worse than that of **MeTiS** on two benchmarks, we still obtain 27.9% average MinCut improvement and between -3.6% and 71.5% improvement in MinCut. All evaluations that twenty runs of two algorithms on 18 graphs are run on an 1800MHz AMD Athlon2200 with 512M memory and can be done in two hours.

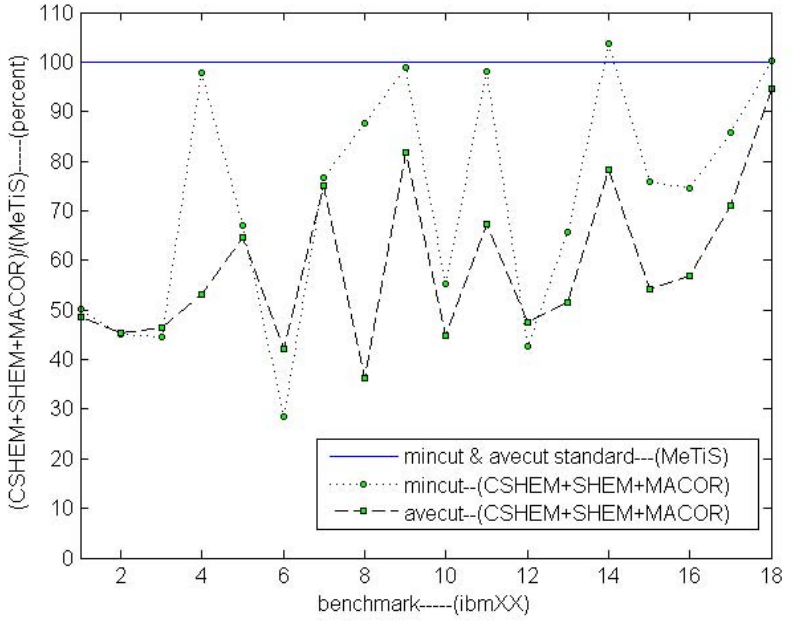

Fig. 1. The MinCut and AveCut comparisons of two algorithms on 18 graphs

6 Conclusions

In this paper, we have presented an effective multi-level algorithm based on ACO. The success of our algorithm relies on exploiting both the ACO method and the concept of the graph core. We obtain excellent *bipartitioning* results compared with those produced by **MeTiS**. Although it has the ability to find cuts that are lower than the result of **MeTiS** in a reasonable time, there are several ways in which this algorithm can be improved. For example, we note that adopting the CSHEM algorithm alone leads to poorer experimental results than

the combination of CSHEM with SHEM. We need to find the reason behind it and develop a better matching-based coarsening scheme coupled with MACOR. In the MinCut evaluation of benchmark ibm14 and ibm18, our algorithm is 3.6% worse than **MeTiS**. Therefore, the second question is to guarantee find good approximate solutions by setting optimal set of parameters values for MACOR.

Acknowledgments

This work was supported by the international cooperation project of Ministry of Science and Technology of PR China, grant No. CB 7-2-01, and by "SEC E-Institute: Shanghai High Institutions Grid" project. Meanwhile, the authors would like to thank professor Karypis of University of Minnesota for supplying source code of **MeTiS**. The authors also would like to thank Alpert of IBM Austin Research Laboratory for supplying the ISPD98 benchmark suite.

References

1. Zha, H., He, X., Ding, C., Simon, H., Gu, M.: Bipartite graph partitioning and data clustering. Proc. ACM Conf Information and Knowledge Management (2001) 25–32
2. Ding, C., He, X., Zha, H., Gu, M., Simon, H.: A Min-Max cut algorithm for graph partitioning and data clustering. Proc. IEEE Conf Data Mining (2001) 107–114
3. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. WH Freeman, New York (1979)
4. Bui, T., Leland, C.: Finding good approximate vertex and edge partitions is NP-hard. Information Processing Letters, Vol. 42 (1992) 153–159
5. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal, Vol. 49 (1970) 291–307
6. Fiduccia, C., Mattheyses, R.: A linear-time heuristics for improving network partitions. Proc. 19th Design Automation Conf (1982) 175–181
7. Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning. Integration, the VLSI Journal, Vol. 19 (1995) 1–81
8. Tao, L., Zhao, Y.C., Thulasiraman, K., Swamy, M.N.S.: Simulated annealing and tabu search algorithms for multiway graph partition. Journal of Circuits, Systems and Computers (1992) 159–185
9. Kadłuczka, P., Wala, K.: Tabu search and genetic algorithms for the generalized graph partitioning problem. Control and Cybernetics (1995) 459–476
10. Żola, J., Wyrzykowski, R.: Application of genetic algorithm for mesh partitioning. Proc. Workshop on Parallel Numerics (2000) 209-217
11. Bahreininejad, A., Topping, B.H.V., Khan, A.I.: Finite element mesh partitioning using neural networks. Advances in Engineering Software (1996) 103–115
12. Leng, M., Yu, S., Chen, Y.: An effective refinement algorithm based on multi-level paradigm for graph bipartitioning. The IFIP TC5 International Conference on Knowledge Enterprise, IFIP Series, Springer (2006) 294–303
13. Leng, M., Yu, S.: An effective multi-level algorithm for bisecting graph. The 2nd International Conference on Advanced Data Mining and Applications, Lecture Notes in Artifical Intelligence Series, Springer-Verlag (2006) 493–500

14. Karypis, G., Kumar, V.: MeTiS 4.0: Unstructured graphs partitioning and sparse matrix ordering system. Technical Report, Department of Computer Science, University of Minnesota (1998)
15. Selvakkumaran, N., Karypis, G.: Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization. IEEE Trans. Computer Aided Design, Vol. 25 (2006) 504–517
16. Amine, A.B., Karypis, G.: Multi-level algorithms for partitioning power-law graphs. Technical Report, Department of Computer Science, University of Minnesota (2005) Available on the WWW at URL http://www.cs.umn.edu/~metis
17. Korošec, P., Šilc, J., Robič, B.: Solving the mesh-partitioning problem with an ant-colony algorithm, Parallel Computing (2004) 785–801
18. Alpert, C.J.: The ISPD98 circuit benchmark suite. Proc. Intel Symposium of Physical Design (1998) 80–85
19. Dorigo, M., Gambardella.,L.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation (1997) 53–66
20. Dorigo, M., Maniezzo, V., Colorni., A.: Ant system: Optimization by a colony of cooperating agents. IEEE Trans on SMC (1996) 29–41
21. Langham, A.E., Grant, P.W.: Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies. Advances in Artificial Life, Lecture Notes in Computer Science Series, Springer-Verlag (1999) 621–625
22. Seidman, S.B.: Network structure and minimum degree. Social Networks (1983) 269–287
23. Batagelj, V., Zaveršnik, M.: An $O(m)$ Algorithm for cores decomposition of networks. Journal of the ACM (2001) 799–804
24. Batagelj, V., Zaveršnik, M.: Generalized cores. Journal of the ACM (2002) 1–8