Cutting Graphs Using Competing Ant Colonies and an Edge Clustering Heuristic

Max Hinne and Elena Marchiori

Radboud University Nijmegen

Abstract. We investigate the usage of Ant Colony Optimization to detect balanced graph cuts. In order to do so we develop an algorithm based on *competing* ant colonies. We use a heuristic from social network analysis called the *edge clustering coefficient*, which greatly helps our colonies in local search. The algorithm is able to detect cuts that correspond very well to known cuts on small real-world networks. Also, with the correct parameter balance, our algorithm often outperforms the traditional Kernighan-Lin algorithm for graph partitioning with equal running time complexity. On larger networks, our algorithm is able to obtain low cut sizes, but at the cost of a balanced partition.

1 Introduction

Networks are quickly becoming one of the most well-studied data structures. Many interesting phenomena can intuitively be seen as networks. Some examples include the World Wide Web, social networks, neural networks, traffic networks and gene regulatory networks. Unfortunately, many of the most interesting data sets are extremely large; several billion nodes are no exception. Even with the advances in modern hardware, this severely hampers analysis of such huge networks. One of the consequences is that research on these networks must employ efficient algorithms, since even algorithms with polynomial complexity can be prohibitively expensive.

We consider the network as a graph consisting of a set of vertices and edges. In this paper we study one particular well-known problem which is finding a minimum cut, i.e. the division of the set of vertices into two disjoint subsets with the smallest possible number of intra-set edges. This problem has many applications, for example in computer vision and network design, and also forms the basis of several network clustering techniques that operate by repeatedly bisecting a graph [1]. In particular in the latter example, the minimum cuts are subject to an additional requirement that captures the relative size of the subsets. In practice, it is often desirable that these are balanced, i.e. they are of roughly the same size. This is not reflected in the minimum cut concept itself. Several measures have been suggested that do take this balance into account. We will consider *conductance* [2]. Minimizing the conductance of a network has been shown to be a problem that is NP-complete [3].

Instead of calculating the minimum conductance exactly, we will apply a heuristic approach to finding an optimal graph cut based on the Ant Colony

Optimization [4] meta-heuristic. We will show that this method is able to find graph cuts on several real-world data sets.

The problem of finding the minimum cut of a graph is well-studied. Originally the problem was solved by dividing it into an easier problem, which was to find a *minimum* $s-t$ *cut*. This entailed finding a cut so that vertices s and t became disconnected. Repeated application of a minimum $s-t$ cut algorithm on all pairs $(s,t) \in V$ yields the minimum cut. Several approaches have been suggested that can reduce the running time of such algorithms. For example, Hao and Orlin [5] devised an algorithm that has a running time of $O(m(n-\lambda)\log(1/\epsilon))$ on a graph with n vertices, m edges and λ the node connectivity[1]. The problem of finding a minimum cut has been shown to be equivalent to the problem of finding the maximum flow of a network. Consequently, algorithms that solve the latter can also be used to solve the former. An extensive overview of available algorithms is given in [6]. Most of these algorithms have running time complexities of $O(nm^2)$, $O(n^3)$ or $O(mn^2)$, which makes them all computationally quite expensive on large graphs. When the problem is not just finding a minimum cut, but finding a *balanced* minimum cut – as captured by conductance – the task becomes harder. Essentially, all possible cuts should be considered, but as this number increases exponentially with n, this leads to an intractable problem [3]. In order to be able to use conductance, poly-logarithmic approximation algorithms are used based on spectral analysis of the adjacency matrix that corresponds to the graph [7], or heuristics [8]. Although such algorithms have a poly-logarithmic running time, calculating the spectrum of a matrix can still be computationally expensive. This leaves room for improvement, which we attempt through the Ant Colony Optimization (ACO) meta-heuristic. A number of interesting methods for graph partitioning problems based on ACO have been introduced, e.g., [9–11]. In this paper we adopt an approach based on competing ant colonies [10]: two colonies compete for resources and reconstruct a global environment corresponding to a good graph partition. Our algorithm differs from previous methods based on competing ACO mainly in the type of heuristic it uses, which is used in social network analysis.

2 Theoretical Background

2.1 Graph Cuts

Let $G = (V, E)$, $E \subseteq V \times V$, be a graph with $n = |V|$ vertices and $m = |E|$ edges. In this paper, we consider only *undirected*, *unweighted* graphs, but our methodology can easily be extended to graphs with edge directions and weights.

A *cut* is defined as a partition of the (vertices of the) graph into two disjoint subsets, S and $V \setminus S = \overline{S}$. The associated *cut-set* is the set of edges for which one end point is in one element of the partition and the other end point in the

[1] The size of the smallest number of nodes that must be removed in order to leave the graph disconnected. This can be calculated a priori in $O(mn)$ time.

other element. The *size* of the cut is the number of edges in the cut set, defined as

$$c_G(S) \equiv \left|\left\{\{u,v\} \in E \,|\, u \in S, v \in \overline{S}\right\}\right| . \tag{1}$$

We omit the subscript G when there is no confusion likely to occur.

The problem we consider is that of finding a *balanced cut*, i.e. the cut for which the cut-set is smaller than any other cut possible on the graph, while at the same time balances the sizes of the two sets of the partition. This is reflected in the *conductance* [7, 12] measure, which is defined as:

$$\phi(S) \equiv \frac{c(S)}{\min(vol(S), vol(\overline{S}))} , \tag{2}$$

where $c(S)$ is the size of the cut-set, and $vol(S)$ is the volume of S, i.e. the number of edges that have both end points within S:

$$vol(S) \equiv |\{\{u,v\} \in E \,|\, u, \in S, v \in S\}| . \tag{3}$$

Note that a lower conductance indicates a better balance, given the same size of the cut-set. The conductance of the whole graph G is the minimum conductance over all possible cuts:

$$\Phi(G) \equiv \min_{S \subseteq V} \phi(S) . \tag{4}$$

Finding the minimum conductance of a graph has been shown to be NP-complete [3], which is why we resort to a probabilistic meta-heuristic.

2.2 Ant Colony Optimization

Ant Colony Optimization (ACO) is an approach to tackle combinatorial optimization problems (e.g. dividing the vertex set into an optimal partition according to some quality function) [4]. ACO considers *solution components* C and a *pheromone model* T. Several independent agents construct a solution s to the problem by combining elements from the solution components. These components $c_i \in C$ have pheromone values $\tau_i \in T$ assigned to them that are used to obtain the probability that the ant moves to the next component, e.g.

$$\Pr(c_i|s) \propto \frac{\tau_i}{\sum_{c_j \in N(s)} \tau_j} , \tag{5}$$

with $N(s)$ the options available to the ant. In Section 3 we will go deeper into this probability distribution.

The general way to solve a combinatorial optimization problem using ACO is to

1. assemble a candidate solution from components based on the probabilities of the pheromone model,

Table 1: The outline of the ACO graph cutting algorithm.

```
Algorithm 1: Cutting graphs through ACO.
Cut GraphCut()
    C = LoadGraph();
    T = InitializePheromones();
    H = InitializeHeuristics();
    Cut s = (V, ∅);
    while no convergence do
        s' = GenerateSolution(C, T, H);
        if φ(s') < φ(s) then
            s = s';
            UpdatePheromones(T, H, s);
        fi;
    od;
    return s;
```

2. update the pheromone model based on the quality of the candidate solution and
3. repeat the first two steps until a satisfying solution is obtained. [13]

In the next section we describe how we translate the cutting of graphs into the ACO paradigm.

3 The Algorithm

The general ACO graph cutting algorithm is provided in Table 1 [4]. Its implementation depends on the two subroutines GenerateSolution(C, T, H) and UpdatePheromones(T, H, s).

In case of a graph cut algorithm, a solution is obviously a partition of the graph. To enable the construction of candidate solutions, we deviate from traditional ACO based on the ant colony metaphor. Instead, we use two *competing* ant colonies. The intuition behind this idea is that both colonies of ants will try to obtain a densely connected subset of the graph, that is only sparsely connected to the subset belonging to the other ant colony. Together, the sets of vertices controlled by the ant colonies correspond to a graph cut. In this scenario, the solution components are the edges in the graph. Whenever an ant colony sends an ant to traverse an edge and marks the target vertex, that edge becomes a part of one of the two components of the cut.

Based on the outcome of several of such 'ant colony competitions', the cut with the lowest conductance is selected as the candidate solution s (see Table 2). Afterwards, pheromones are deposited on edges *within* S and $\overline{S}$, so that in subsequent rounds the ants will favor walking along these edges. Eventually the process is ended when a fixed number of iterations is completed.

The CutGraph(C, T, H) function, where the actual ant colony competition takes place, is described separately in Table 3. In the initialization function, the

Table 2: The GenerateSolution(C, T, H) subroutine.

```
Algorithm 2: Generating a solution.
Cut GenerateSolution(C, T, H)
    S = ∅;
    while i < k do
        s = CutGraph(C, T, H);
        S = S ∪ s;
        i = i + 1;
    od;
    return argmin_{s∈S} φ(s);
```

Table 3: The CutGraph(C, T, H) subroutine.

```
Algorithm 3: Cutting a graph.
Cut CutGraph(C, T, H)
    A = InitializeColonies(Colony 1, Colony 2);
    while not all vertices flagged do
        foreach colony a ∈ A do
            e = SelectEdge(C, T, H);
            a.MoveAntAcross(e);
            a.FlagVertex(a);
        od;
    od;
```

ant colonies are given random starting vertices. The ACO paradigm comes into play in the SelectEdge(C, T, H) function, where a colony must decide to which neighboring vertex it sends an ant next. Out of the possible edges $N(s)$ the colony makes its selection in a probabilistic manner. The probability of an edge $e_i \in N(s)$ is given by

$$\Pr(e_i|s) \equiv \frac{\tau_i^\alpha \eta(e_i)^\beta}{\sum_{e_j \in N(s)} \tau_j^\alpha \eta(e_j)^\beta}\,, \tag{6}$$

with τ_i the pheromone associated with e_i and $\eta(e_i)$ represents optional prior knowledge, or *heuristic information*, the colonies have about the attractiveness of e_i (more on this later). The parameters $\alpha \geq 0$ and $\beta \geq 0$ determine the relation between pheromone information and heuristic information, respectively.

The final part of the ACO approach is the updating of the pheromone distribution, so that solution components from successful solution candidates are more likely to occur in future solution candidates. This is accomplished in UpdatePheromones(T, H, s). In this procedure, pheromones slowly evaporate over time as well. This is done to prevent the colonies from getting stuck in local optima. For each $\tau_i \in T$, τ_i is updated according to

$$\tau_i = \rho\tau_i + \delta_i(s) f(s)\,, \tag{7}$$

with ρ the parameter representing the evaporation rate ($\rho = 0.1$ in our experiments), $\delta_i(s)$ is 1 if $e_i \in s$ (i.e. the end points are either both in S or both in $\overline{S}$), 0 otherwise and finally $f : \mathcal{S} \to \mathbb{R}$ is a function that maps a solution to a score, using $f(s) \equiv c\frac{1}{\phi(s)}$, where $\phi(s)$ is the conductance corresponding to this cut and c a scaling constant. For now, we simply choose $c = 1$.

3.1 Heuristic Information

Ants are not generally considered bright individuals, but in the ACO paradigm there is room for heuristic information. This represents the idea that ants (or their collectives) do not base their decisions on pheromone information only, but also on some local characteristic of the network connectivity. In Eq. (6) this is reflected in the parameter $\eta(e_i)$.

In the case of finding a balanced graph cut, the heuristic information should reward edges that are likely to be inter-set edges. We borrow a measure from social network analysis that is used in community detection, namely the *edge-clustering coefficient*. The coefficient $C(e)$ of an edge [14], which counts the number of triangles that an edge is part of, is expressed as:

$$\eta(e_i) \equiv C(e_i) = \frac{|Nb(s) \cap Nb(t)|}{\min\left(d(s) - 1), (d(t) - 1\right)} + 1\,. \tag{8}$$

with s and t the end points of e_i, $Nb(s)$ the neighbors of a vertex s and $d(s) = |N(s)|$ the degree of a vertex. The $+1$ in the denominator prevents the heuristic distribution becoming 0, which would make it impossible for the edge to be selected at all.

3.2 Stop Conditions

Although by the nature of the ACO paradigm, the algorithm converges, it is out of the scope of this study to derive the exact time complexity in terms of the approximation of the true minimal conductance (e.g. [15]). Therefore we settle for one of two possible stop conditions for the outer loop of the algorithm (see Table 1). The first alternative is to simply execute a fixed number of iterations of $\mathsf{GenerateSolutions}(C, T, H)$. For small graphs, this gives satisfactory results (see Section 4).

For larger graphs, a threshold γ can be used to indicate the minimum improvement of Φ between each iteration. If $\Delta\Phi$ drops below γ, the algorithm terminates.

4 Validation

To test the performance of our algorithm, conducted a series of experiments on different types of networks. In the first series, we executed the ant colony graph cutting algorithm on two real-world networks for which a ground truth is known.

The first network we consider is the famous karate club network [16]. The network has become a staple example in literature on clustering/cutting. It consists of the social network of 34 members of a US university karate club. The 78 edges in the network represent friendships between the members. The network was followed by Zachary for a period of two years, during which the club split after a dispute. The split caused the club to break into two new clubs, one centered around the former instructor and one around one of the members. The separation into these groups is often taken as the ground truth division (i.e. cut) of this network [17]. The conductance of the actual cut on the karate club network is $\Phi = 0.30$. We obtained a cut through our ACO algorithm as well, using the following settings: 25 iterations, 10 trials per iteration, $\alpha = 1$ and $\beta = 0$ (i.e. heuristic information was not used). The result of this cut is shown in Figure 1. As can be seen, our cut corresponds very well to the actual division, with only node 10 being classified on the wrong side of the cut (this corresponds to an accuracy of 0.97). The minimum conductance we obtained is actually lower than the ground truth, $\Phi = 0.29$.

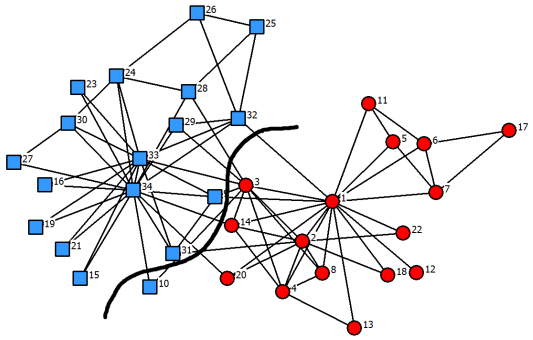

Fig. 1: The karate club network. The shape of the nodes indicate the partitioning in the actual network. The cut obtained by our algorithm (for parameters, see text) is given by the bold line.

The second data set is a network of books sold by Amazon.com about US politics published around the presidential elections of 2004. The 105 nodes in this network are the individual books, the 374 edges represent frequent co-purchasing by customers of Amazon. The data was collected by Krebs [18]. Later, the ground truth labels of the books were determined by Newman by looking at the book descriptions[2]. The books were classified into 'liberal', 'neutral' and 'conservative'. Since we wanted to test our algorithm on a binary data set, we removed the five books that were labeled 'neutral'. With the remaining sets of liberal and conservative books, the cut has a conductance of $\Phi = 0.07$. Using the same settings as for the karate club network, our algorithm identified a cut with conductance

[2] See `http://www-personal.umich.edu/~mejn/netdata/`

$\Phi = 0.04$. The accuracy of this cut was 0.96, indicating that the obtained cut nearly coincides with the actual division.

4.1 Parameter Tuning

For many other networks, a ground truth is generally not available. To gain insight in the performance of our algorithm nonetheless, we executed it on the neural network of the roundworm *C. Elegans* [19], which consists of 297 vertices and 2148 edges. We considered the conductance of the optimal cut after an increasing number of iterations of the GenerateSolution(C, T, H) procedure. We used 10 solutions per iteration and up to 25 iterations. The first experiment considers the impact of using just pheromones ($\alpha = 1$, $\beta = 0$), just heuristic information ($\alpha = 0$, $\beta = 1$), both in equal proportions ($\alpha = 1$, $\beta = 1$) and none ($\alpha = 0$, $\beta = 0$), as a baseline to compare the performance. In the baseline, the edge selection process is essentially random. The experiment was repeated 20 times. Figure 2a shows the mean conductance as a function of the number of iterations. The chart shows each parameter setting outperforms the baseline, however, only by a small margin when only heuristic information is used. Using pheromones and heuristic information combined, or just pheromones alone, leads to significantly better cuts than the baseline settings.

The second experiment considers using different ratios between α and β. Again, we used 10 solutions per iteration and up to 25 iterations, and repeated each experiment 20 times. This time however, we consider strong favor for pheromones ($\alpha = 10$, $\beta = 1$), strong favor for heuristic information ($\alpha = 1$, $\beta = 10$), strong emphasis on both ($\alpha = 10$, $\beta = 10$) and again the baseline ($\alpha = 0$, $\beta = 0$). It is important to note that for a pheromone τ_i, $\tau_i \in [1, \infty)$. For heuristic information η, $\eta \in [1, 2]$. Consequently, the results for parameter settings $\alpha = 1$, $\beta = 1$ behave differently than $\alpha = 10$, $\beta = 10$. The results from this experiment are shown in Figure 2b. The results show that too much emphasis on the pheromone information may lead to bad cuts, which we attribute to getting stuck in local optima. The best performance is obtained when heuristic information is weighed heavily, in combination with a little guidance by the pheromone distribution.

The experiments show that the two defining features of the ACO paradigm, pheromone- and heuristics based selection of solution components, indeed lead to better graph cuts. However, the heuristic information is far more crucial than the pheromones, although the latter is needed to improve performance in subsequent iterations.

4.2 Comparison to a Baseline Algorithm

In order to further analyze the performance of our algorithm, we compare it to a well-known graph partitioning algorithm as proposed by Kernighan and Lin [8]. The algorithm tries to minimize the cut set while balancing the partition elements by repeatedly swapping vertices from the sets. It has a running time

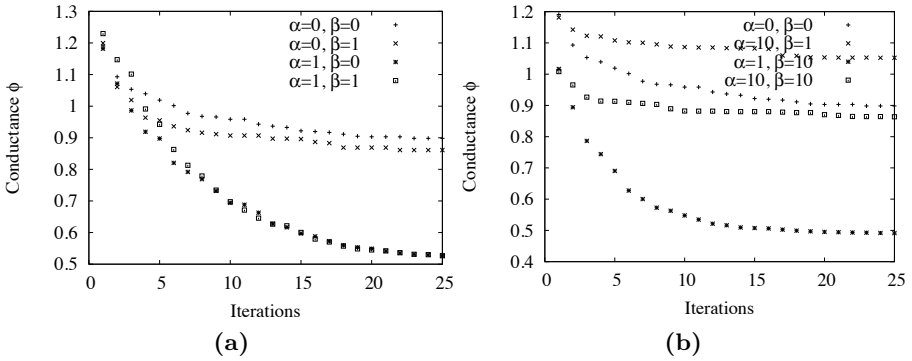

Fig. 2: The mean conductance ϕ of the optimal solution after I iterations (repeated 20 times), for different settings of α and β (see text). Error bars have been omitted for clarity.

complexity of $O(n^2 \log n)$ [20]. We compared the cuts obtained by the Kernighan-Lin algorithm to our own approach, using 25 iterations of 10 solutions, with $\alpha = 1$ and $\beta = 10$. The datasets we used were all taken from the online collection of Mark Newman[3].

The mean results of 20 runs of this experiment are shown in Table 4. As the conductance scores show, our algorithm obtains better cuts in 5 of 6 datasets, and obtains the same cut in one network. Only once is the algorithm outperformed by the baseline algorithm. The table also shows the sizes of the cut-set, which is used as an alternative for conductance when both partitions are required to be equal in size.

Table 4: Conductance and cut size for the Kernighan-Lin algorithm (KL) and the mean conductance for the ant colony optimization algorithm (ACO), for several network datasets. n and m denote the number of vertices and edges in the networks, respectively. The number in parenthesis is the standard deviation.

Network	n	m	Φ_{KL}	c_{KL}	Φ_{ACO}	c_{ACO}
Zachary karate club [16]	34	78	0.64	36	**0.29** (0)	**10** (0)
Bottlenose dolphins [21]	62	159	0.36	46	**0.21** (0.04)	**11.15** (3.54)
C. Elegans neural network [19]	297	2345	0.49	716	0.49 (0.02)	**402** (20.37)
Football players [22]	115	613	0.37	190	**0.25** (0.03)	**67.75** (7.81)
Political books [18]	105	441	**0.04**	**16**	0.09 (0)	19 (0)
Les Miserables [23]	77	254	0.35	56	**0.29** (0.01)	**23** (12)

[3] `http://www-personal.umich.edu/~mejn/netdata/`

4.3 Complexity

At each step in the cutting of a graph, our algorithm considers at most m edges and selects the optimal to traverse according to Eq. (6). The last calculation is done in $O(1)$. The selection of edges is repeated until all n vertices in the network have been flagged. As this process is repeated for each iteration (of a total of I) and each trial (a total of S per iteration), the total running time complexity of the algorithm is $O(ISnm)$. In general, we assume that $I \ll n$ and $S \ll n$, so that the complexity may be considered $O(nm)$. If we furthermore assume that we the algorithm is run on sparse networks, i.e. $m = O(n)$, our algorithm has running time complexity like the Kernighan-Lin algorithm.

4.4 Comparison to State-of-the-Art Cuts

As our final experiment, we compared the cuts of our algorithm to the best partitions known of five larger datasets. The networks and the best known cuts were taken from the University of Greenwhich Graph Partitioning Archive[4] (GPA). Unfortunately, these cuts do not list the corresponding conductance values. Instead, they show the cut size and the size of the largest element of the partition. A further difference with our approach is that the cuts from the archive are all accompanied by a known balance score B, defined as

$$B = \frac{\max\left(|S|, |\overline{S}|\right)}{n/2} \quad , \tag{9}$$

with n the number of vertices in the graph. Although the optimization of conductance favors balanced partitions, it is possible that the ACO algorithm yields an imbalanced partition. Therefore, we compare the performance of our algorithm to the most lenient cut known from the archive, those with balance up to $B \leq 1.05$. Table 5 shows the results for the first five datasets in the archive. As the numbers indicate, the ACO algorithm is not (yet) up to par with state-of-the-art algorithms on larger graphs. Although the algorithm is sometimes able to detect a cut with a cut size lower than the best known, it does so with a very imbalanced partition.

5 Conclusion and Suggestions for Further Research

In this paper we have explored the usage of the Ant Colony Optimization paradigm to find balanced minimal graph cuts. We developed an algorithm based on the general ACO outline that uses ant systems to build a solution to the problem based on individual components (graph edges). Crucial to our approach – and different from most conventional ACO algorithms – is that we use two *competing* ant colonies. Both colonies try to claim as much of the graph as possible. The resulting front line corresponds to the graph cut of a single iteration of the algorithm.

[4] `http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/`

Table 5: Mean cut size for 20 repetitions of the Ant Colony Optimization (ACO) algorithm, the best known cut from the Graph Partitioning Archive (GPA), and the best cut and corresponding balance obtained by ACO. The numbers within parentheses indicate one standard deviation.

Network	n	m	c_{ACO}	B_{ACO}	c_{GPA}	$c_{\text{ACO}_{best}}$	$B_{\text{ACO}_{best}}$
add20	2395	7462	611.24 (308.31)	1.26 (0.29)	**550**	141	1.74
data	2851	15093	**78.12** (13.19)	1.57 (0.02)	181	61	1.54
3elt	4720	13722	131.50 (12.60)	1.03 (0.05)	**87**	113	1.02
uk	4824	6837	**7.00** (1.49)	1.56 (0.04)	18	4	1.56
add32	4960	9462	**7.10** (0.31)	1.12 (0.02)	10	7	1.12

Using the traditional methods to update pheromones along solution components that are part of successful solutions, as well as the use of a heuristic based on edge clustering, our algorithm is able to obtain cuts with low conductance. Our experiments show that strong emphasis on the heuristic information speeds up the detection of the optimal cut. Furthermore, we have shown that cuts on small networks correspond very well to the true cut. This suggests that our algorithm can be used as a feasible approximation of an otherwise intractable problem. When comparing our algorithm to benchmark and state-of-the-art algorithms, we observed that the ACO algorithm is able to obtain cuts with lower conductance than the Kernighan-Lin algorithm. However, the ACO algorithm delivers rather unbalanced cuts on larger networks. This shows that although the approach certainly has potential, more work is needed to enforce more balanced solutions.

In further research, we intend to experiment with the optimization of cut size alone. Also, more sophisticated techniques may be used to initialize the ant colonies on the networks, so that local optima can be avoided easier. Lastly, ACO uses several heuristic constants, such as the evaporation rate of the pheromones, the relation between conductance and pheromone updates and the factors α and β that tune the ratio between pheromones and heuristic information. For each of these parameters, optima should be identified.

Acknowledgments We wish to thank Mart Gerrits and Bob van der Linden for their contributions to the experimentation software and the anonymous reviewers for their constructive comments on Section 4.

References

1. G.W. Flake, R.E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1:385–408, 2004.
2. F.R.K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, February 1997.
3. J. Síma and S.E. Schaeffer. On the NP-Completeness of Some Graph Cluster Measures. *CoRR*, abs/cs/0506100, 2005.

4. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41, 1996.
5. J. Hao and J.B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994.
6. A.V. Goldberg and R.E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.
7. U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *In 11th Europ. Symp. Algorithms*, pages 568–579. Springer-Verlag, 2003.
8. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(1):291–307, 1970.
9. P. Korošec and J. Šilc. Multilevel optimization of graph bisection with pheromonoes. In Bogdan Filipič and Jurij Šilc, editors, *Proceedings of the International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004)*, pages 73–80, Ljubljana, Slovenia, October 11-12 2004. Jožef Stefan Institute.
10. A.E. Langham and P.W. Grant. Using Competing Ant Colonies to Solve k-way Partitioning Problems with Foraging and raiding strategies. In D. Floreano, J.D. Nicoud, and F. Mondana, editors, *Proceedings of the Fifth European Conference on Artificial Life (ECAL)*, pages 621–625. Springer-Verlag, Heildelberg, Germany, 1999.
11. M. Leng and S. Yu. An effective multi-level algorithm based on ant colony optimization for bisecting graph. In *PAKDD*, volume 4426 of *LNCS*, pages 138–149. Springer, 2007.
12. D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. *ACM Trans. Database Syst.*, 31(4):1499–1525, December 2006.
13. C. Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373, December 2005.
14. S. Papadopoulos, A. Skusa, A. Vakali, Y. Kompatsiaris, and N. Wagner. Bridge bounding: A local approach for efficient community discovery in complex networks. Technical report, Informatics & Telematics Institute (CERTH), 2009.
15. W.J. Gutjahr. First steps to the runtime complexity analysis of ant colony optimization. *Comput. Oper. Res.*, 35(9):2711–2727, 2008.
16. W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.
17. M. E. J. Newman. Fast algorithm for detecting community structure in networks. Sep 2003.
18. V. Krebs. Political polarization during the 2008 us presidential campaign.
19. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.
20. C. P. Ravikumar. *Parallel Methods for VLSI Layout Design*. Greenwood Publishing Group Inc., Westport, CT, USA, 1995.
21. D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, and S.M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.
22. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12):7821–7826, June 2002.
23. D. E. Knuth. The stanford graphbase: A platform for combinatorial computing. Addison-Wesley, Reading, MA, 1993.