Efficient Implementation of Community Detection Using Extremal Optimization

Jan Varho

August 25, 2009

1 Introduction

Community structure is a feature of complex networks, describing the modular hierarchy found in many real networks. Informally, communities in networks are understood to be densely connected subgraphs, with fewer links to nodes outside the community.

Modularity as a measure of community structure was first proposed by Newman and Girvan [1]. Subsequently, many successful community detection algorithms have been based on modularity optimization [2, 3, 4]. While there are problems with this approach to community detection [5, 6], modularity optimization can be adapted to overcome them [7] and remains useful.

Since the problem of modularity optimization is NP-complete [8], there is probably no polynomial time algorithm to find the exact solution. Brute force would take at least exponential time, so a heuristic approach has to be used instead.

One effective optimization heuristic is *extremal optimization*, which was developed by Boettcher and Percus [9] and applied to modularity optimization by Duch and Arenas [3]. Here a modified EO algorithm with a lower computational cost is used instead. The streamlined algorithm has order $\mathrm{O}\left(n \log^3 n\right)$ compared to $\mathrm{O}\left(n^2 \log^2 n\right)$ of the DA algorithm.

The algorithm is explained in Section 2 and its implementation described in Section 3, which also includes complexity analysis. While the description is given for an unweighted, undirected graph, it can readily be generalized to weighted and directed cases. In Section 4, the results of running the algorithm on existing and new data are presented.

2 Algorithm

Extremal optimization is a heuristic for optimizing a global variable, known as the *benefit function*, using a local one. It repeatedly removes the worst components of a partial solution in search of a better one. In this case, the benefit function is the modularity of [1], which can be expressed as

$$Q = \sum_r q_r = \sum_r \kappa_r/K - k_r^2/K^2 = \sum_i \kappa_i/K - k_i k_{r(i)}/K^2, \tag{1}$$

where k_i is the number of links for node i and κ_i the number of links to nodes in the same community. The semi-global quantities k_r and κ_r are defined as sums over the nodes in community r. The global $K = \sum_i k_i = 2m$ is constant.

The local variable given in [3] is the normalized modularity contribution of a single node:

$$\lambda_i = \kappa_i/k_i - k_{r(i)}/K, \tag{2}$$

where the second term is shared by all the nodes in a community. This is called the *fitness* of node i.

The algorithm itself is stochastic and divisive, recursively splitting a partition into two, until further partitioning would decrease the global modularity. Each partitioning of nodes $i = 1, 2, \ldots, n$ begins with a random split into r_1 and r_2 and continues in four steps:

1. Order the nodes by fitness, such that $\lambda_i < \lambda_{i+1}$.
2. Pick the node l with probability $P(l) \sim l^{-\tau}$.
3. Move node l to the other partition.
4. If current modularity $q_{r_1} + q_{r_2}$ is the highest so far, record this state.

Steps 1–4 are repeated until the maximal modularity remains unchanged for αn steps, with α a constant. Then the resulting two partitions are kept, unless the modularity is lower than that for a single undivided partition.

It can be shown that there is an optimal value for the exponent: with the above choices, $\tau \sim 1 + \log\alpha/\log n$ approximates the asymptotic case [9]. For huge networks it may be beneficial to choose a lower value, but for networks of up to tens of thousands of nodes, values between 1.4 and 1.7 gave results close to optimal. The value $\tau = 1.5$ was used to obtain the results in Section 4.

Unlike in the DA version of the algorithm, even if a partition contains disconnected components, it is considered a single community for intermediate modularity calculations. This allows for some optimizations in implementation.

3 Implementation

Since the second term of fitness in Equation 2 is the same for all nodes in community r, ordering nodes $i \in r$ by λ_i is equivalent to ordering them by κ_i/k_i. Moving a node between two partitions then involves updating the κ_i/k_i values for the up to k_i connected nodes. Thus, maintaining heap data structures for each partition leads to a total $\mathrm{O}(k_i \log n)$ time for steps 1 and 3.

Updating the semi-global k_r and κ_r involves only subtraction and addition of local variables, as can bee seen from Equation 1. The best state can be recorded by maintaining an array of moves since that state. Step 4 then takes $\mathrm{O}(1)$ time.

With separate heaps for the two partitions, the node with the worst fitness can be found in constant time by comparing fitnesses for the first node of each heap. The lth node could be found by iteratively removing the first one and then replacing the nodes afterwards. This would take $\mathrm{O}(\langle l \rangle \log n)$ time. However, accepting a partial heap order for the nodes, like in [9], step 2 can be completed in constant time for any l using a simple array access.

Therefore, a single iteration of the algorithm can be done in $\mathrm{O}(\langle k \rangle \log n)$ time. The expected number of iterations required for a single split using $\alpha = 1$ is $\mathrm{O}(n \log n)$ [3]. Assuming an $\mathrm{O}(\log n)$ average depth for the dendogram, the total computational cost is thus $\mathrm{O}\left(m \log^3 n\right)$, or $\mathrm{O}\left(n \log^3 n\right)$ for a sparse graph.

The graph itself is stored as an adjacency list, so that linked nodes can be quickly iterated. Hence, all the data structures used – heaps, lists and arrays – have memory requirements linear in problem size.

Further improvements

In practice, the algorithm is fast enough that higher values of α or the best result of multiple runs can be used for improved accuracy. As α has a predictable, nearly linear impact on running time, it is not a tuning parameter like found in some algorithms. Instead, it can be used to find the most accurate results with the available computational resources.

As a speed optimization, connected components can be moved to separate partitions between splits (outside the inner loop) in linear time. Attempting a connected component search even when a split lowers the modularity of a partition can prevent early termination and lead to better accuracy. Neither optimization changes the computational cost of the algorithm and both were used for the results in the next section.

After the first split, it is possible to handle recursive splits in parallel, e.g. using a multi-core processor. Since individual splits cannot be directly parallelized, there is a limit to parallel scalability. However, both improving accuracy with multiple runs and multi resolution analysis require several independent runs, which can be trivially parallelized.

Network	Nodes	NF	DA	NS	EO
Zachary	34	0.381	0.419	0.419	0.419
Jazz	198	0.438	0.445	0.442	0.442
C. elegans	453	0.400	0.434	0.435	0.436
Email	1133	0.480	0.574	0.572	0.576
PGP	10680	0.733	0.846	0.855	0.874
Cond-Mat	27519	0.668	0.679	0.723	0.731
Running time		$O\left(n \log^2 n\right)$	$O\left(n^2 \log^2 n\right)$	$O\left(n^2 \log n\right)$	$O\left(n \log^3 n\right)$

Table 1: Results for six networks described in the text, using Newman fast (NF), Duch and Arenas (DA), Newman spectral (NS) and the modified extremal optimization (EO) algorithms [2, 3, 4]. Running times are for sparse networks.

4 Data and Results

Comparison

Six networks that have been extensively studied previously were analyzed to verify the comparative accuracy of the algorithm. These were Zachary's karate club network [10], the jazz musicians network [11], the C. elegans metabolic network [3], the URV email network [12], the PGP trust network [13] and the largest connected component of the arXiv.org condensed matter (cond-mat) coauthorship network [14]. These represent networks from tiny to large, though even larger networks have been studied.

Table 1 summarizes the results of running the algorithm on the six networks in question. The results for extremal optimization are best of 10 runs using $\alpha = 100$. The results for the other three algorithms are from the references. Clearly, extremal optimization is competitive in accuracy – even with slower algorithms.

Scaling

Next, synthetic networks of a variable number of nodes were used to show the scalability of the algorithm. The networks were created with four communities of equal size, where nodes were on average connected with ten nodes in the same community and six nodes in other communities.

Since dendogram depth is now independent of network size, the expected scaling is $O\left(n \log^2 n\right)$. Figure 1 shows that the actual results are not too far off. Fluctuations are due to cache and memory effects and the asymptotic nature of dynamic arrays.

Quantum physics network

In addition to the reference networks, a new set of data was gathered. The data is from the quantum physics (quant-ph) list of the arXiv.org eprint archive for the years 1999-2008, downloaded in June 2009. The network consists of

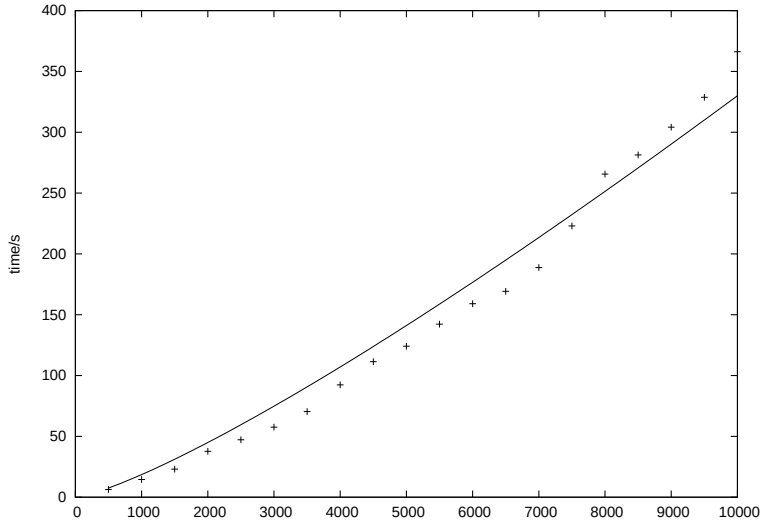

Figure 1: Scaling of the computational cost with network size as explained in text. The times are total processor time over 100 networks with $\alpha = 1$ on a 350 MHz PC. The line is best fit $n \log^2 n$.

Network	Nodes	Links	$\langle k \rangle$	Communities	Modularity
Quant-Ph	14638	51304	7.01	2003	0.739
Largest c.c.	11259	48417	8.60	205	0.736
Community	Largest		Mean	Median	Smallest
Nodes	1120		54.9	12	1

Table 2: The quant-ph coauthorship network, described in text. The number of communities is for maximum modularity over 10 runs using $\alpha = 100$. Community statistics are for those found from the largest connected component.

authors as nodes and coauthorships as unweighted, undirected links. Authors were matched automatically using last name and the first initial of the author, so some duplicates and collisions will certainly exist. An overview of the results is given in Table 2.

The largest connected component was extracted from the total network for separate community analysis. The modularity value is very close to the one for the largest component of the cond-mat network, suggesting a similar degree of hierarchical structure. It is also significantly higher than the modularity of a random scale free network of the same size and connectivity, which according to [15] is 0.352 ± 0.007.

On the other hand, the average community size was lower – 54.9 compared to 61.3 for cond-mat. This is due to the approximate power law distribution of community sizes, and the resolution limit of modularity optimization which also depends on network size [5].

5 Conclusions

In conclusion, extremal optimization is one of the most accurate modularity optimization algorithms that can realistically be used on large networks. It can be implemented with a low computational cost – even competitive with the fastest such algorithms. This makes accurate community analysis of huge networks feasible, even on multiple levels of resolution.

The quant-ph network was found to be similar in structure to the cond-mat network, as expected. A modularity value of 0.736 suggests statistically significant community structure, and community sizes were found to follow a scale free distribution.

Implementing the algorithm for more general definitions of modularity, to allow analysis of directed and weighted networks, requires further work. Thereafter, implementation of the multi-resolution definition in [7] is straightforward. Additionally, a final refinement phase could be implemented to improve the accuracy of results, as has been done with other algorithms [4].

References

[1] M.E.J. Newman and M. Girvan, *Phys. Rev.* E **69**, 026113 (2004).

[2] M.E.J. Newman, *Phys. Rev.* E **69**, 066133 (2004).

[3] J. Duch and A. Arenas, *Phys. Rev.* E **72**, 027104 (2005).

[4] M.E.J. Newman, *Proc. Natl. Acad. Sci. USA* **103**, 8577 (2006).

[5] S. Fortunato and M. Barthélemy, *Proc. Natl. Acad. Sci. USA* **104**, 36 (2007).

[6] J.M. Kumpula, J. Saramäki, K. Kaski and J. Kertész, *Eur. Phys. J.* B **56**, 41 (2007).

[7] A. Arenas, A. Fernández and S. Gómez, *New J. of Phys.* **10**, 053093 (2008).

[8] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski and D. Wagner, *eprint* arXiv:physics/0608255v2 (2006).

[9] S. Boettcher and A.G. Percus, *Phys. Rev.* E **64**, 026114 (2001).

[10] W.W. Zachary, *J. Anthropol. Res.* **33**, 452 (1977).

[11] P.M. Gleiser and L. Danon, *Adv. Complex Syst.* **6**, 565 (2003).

[12] R. Guimerà, L. Danon, A. Diaz-Guilera, F. Giralt and A. Arenas, *Phys. Rev.* E **68**, 065103 (2003).

[13] M. Boguña, R. Pastor-Satorras, A. Diaz-Guilera and A. Arenas, *Phys. Rev.* E **70**, 056122 (2004).

[14] M.E.J. Newman, *Proc. Natl. Acad. Sci. USA* **98**, 404 (2001).

[15] R. Guimerà, M. Sales-Pardo and L.A.N. Amaral, *Phys. Rev.* E **70**, 025101(R) (2004).