A Framework for Fast Community Extraction of Large-Scale Networks

Yutaka I. Leon-Suematsu*
NiCT/ATR
2-2-2 Hikaridai, Seika-cho, Souraku-gun
Kyoto 619-0288, Japan
yutaka@crev.jp

Kikuo Yuta
NiCT/ATR
2-2-2 Hikaridai, Seika-cho, Souraku-gun
Kyoto 619-0288, Japan
yuta@crev.jp

ABSTRACT

Most of the faster community extraction algorithms are based on the Clauset, Newman and Moore (CNM), which is employed for networks with sizes up to 500,000 nodes. The modification proposed by Danon, Diaz and Arenas (DDA) obtains better modularity among CNM and its variations, but there is no improvement in speed as its authors expressed. In this paper, we identify some inefficiencies in the data structure employed by former algorithms. We propose a new framework for the algorithm and a modification of the DDA to make it applicable to large-scale networks. For instance, the community extraction of a network with 1 million nodes and 5 million edges was performed in about 14 minutes in contrast to former CNM that required 45 hours (192 times the former CNM, obtaining better modularity). The scalability of our improvements is shown by applying it to networks with sizes up to 10 million nodes, obtaining the best modularity and execution time compared to the former algorithms.

Categories and Subject Descriptors: H.3.3[Information Search and Retrieval]: Clustering

General Terms: Algorithms, Experimentation

Keywords: Community analysis, Clustering, Large-scale networks

1. INTRODUCTION

The Internet provides services, such as the on-line social-related services namely SNS, allowing interaction in huge networks. One big challenge is developing efficient techniques for identification of highly cohesive subgroups, called communities, for large-scale networks. Clauset, Newman and Moore (CNM) proposed a fast algorithm with $O(n \log^2 n)$[1] that could be applied to networks with sizes of less than 500 thousand nodes. Then, Danon, Diaz and Arenas (DDA)[2] made a modification to the CNM to improve the modularity while retaining its speed. Wakita and Tsurumi (WT)[3] proposed some heuristics to improve the speed of CNM, but with compromises in modularity in their fastest heuristics. All these algorithms are used for undirected networks.

Undirected networks allow representation in symmetric matrices. There are two implementation frameworks, one for CNM and another for WT. Both of them basically use a sparse matrix that stores both symmetric values ΔQ_{ij} and ΔQ_{ji}, and keep them consistent at all times, which impact the performance in both frameworks. For instance, any update in a pair (i, j) takes $O(1)$ but the update of its symmetric pair (j, i) takes $O(\log n)$ because it requires searching column j in row i.

In this paper, we propose a new implementation framework to avoid unnecessary operations to the minimum and a modification of the DDA to fit our framework, obtaining improvements in speed and modularity compared to the former algorithms.

*both authors contributed equally to this work

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

2. FRAMEWORK IMPROVEMENTS

2.1 Data structure improvement

The use of triangular matrices avoids or reduces the previously described implementation difficulties, accelerating the algorithm, because it stores half of the information required in frameworks of CNM and WT due to symmetry.

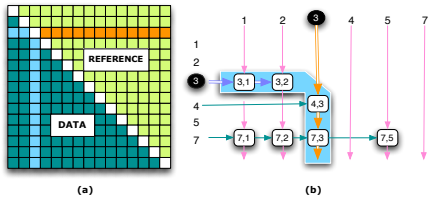

Figure 1: A new framework.

We employ two triangular sparse matrices (Figure 1(a)). The lower triangular matrix stores the values of every ΔQ_{ij} where $i < j$, and the upper triangular matrix stores the references to their symmetric values in the lower triangular matrix. The reference matrix allows access over the columns in the lower triangular matrix in $O(1)$. We use a double-linked sorted list for rows (Figure 1(b)). The advantage is observed in deletions and updates of ΔQ_{ij} which is done in $O(1)$. We keep track of the $\max_j \Delta Q_{ij}$ in row i only for the lower triangular matrix, reducing the search range when obtaining the maximum ΔQ of the row.

In order to iterate over all the neighbors of the community i, or over the whole row in the symmetric matrix, it is necessary to iterate over the elements in the lower triangular matrix and then switch to elements of the upper triangular reference matrix obtaining the position and access of their

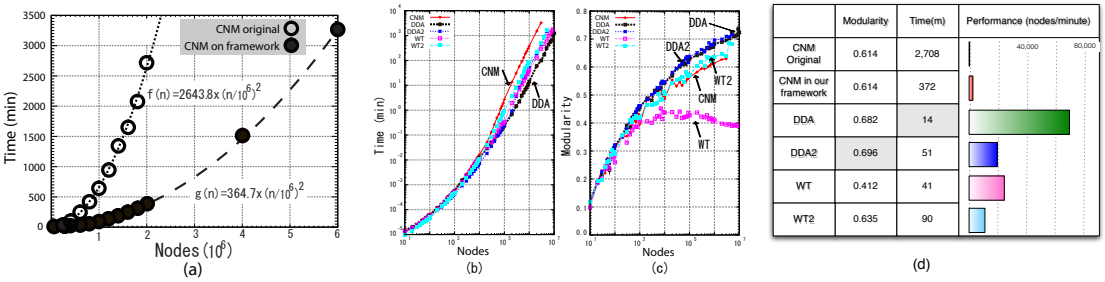

Figure 2: Results from the original CNN and CNM, DDA, DDA2, WT and WT2 under our framework

symmetric cells in the lower triangular matrix (Figure 1(b)). This process is done by the creation of a special *iterator* to switch the matrix when needed.

2.2 Reduction of unnecessary operations

When all values of ΔQ_{ij} in the lower triangular matrix are negative for a certain row i, its maximum value is assigned to a constant negative value in order to avoid unnecessary updates in the max heap H.

In the same way, when combining two in-process communities i and j, we keep track of $\max_k \Delta Q_{jk}$ for the community j. In the case that this value is negative after finishing the combination, any further joining of two communities linked to j will still produce negative ΔQ in the community j. As a consequence, this in-process community will produce no positive value in any circumstances, therefore, we eliminate these rows, reducing unnecessary operations.

2.3 Algorithm improvement

DDA provides better modularity when treating in-process communities of different sizes as similar. DDA normalizes the contribution in modularity by dividing the contribution by the degree of the in-process communities. The main purpose of this modification was to improve the modularity rather than speed as pointed out in [2]. This modification makes the matrix asymmetric, which requires the manipulation of the full matrix. We make a slight modification to the algorithm in order to use it in our triangular matrices.

$$\Delta Q^*_{ij} = \Delta Q_{ij}/a_i \Longrightarrow \Delta Q^*_{ij} = \Delta Q_{ij}/\min(a_i, a_j)$$

This modification will not compromise the result. Where i and j have different degrees and i has the lowest degree, (i, j) will produce a higher value than its symmetric (j, i). Therefore, in any case the higher value (i, j) is preferable.

3. LARGE-SCALE NETWORK

We evaluate our improvements with networks of sizes up to 10 million nodes and 50 million edges generated by the CNNR model[4] that produces SNS-like networks.

We implemented the two most representative heuristics of [3], renaming the HE to WT and HE' to WT2. We created another modification of DDA represented by DDA2 that uses our modified normalization of DDA only in the max ΔQ value of each row, which will be inserted in the max heap H. All the algorithms were implemented in our framework using standard C++ and executed on a PC with Xeon 2.8 GHz CPU, 64GB Ram, Red hat linux. We should note that all programs are single process.

3.1 Experimental Results

The effectiveness of our framework is shown in Figure 2(a). The CNM implemented under our framework is faster than the original CNM, implemented and distributed by Clauset. Two reference lines obtained by fitting the results show that our framework is 7 times faster. Subsequent experiments will employ the CNM implemented under our framework.

The effectiveness and scalability of all the algorithms are shown in Figure 2(b-d). It should be noted that CNM was applied to networks up to 1 million nodes because of limitations for larger networks. Figure 2(b) shows the execution time, where all modifications of CNM are faster than CNM, our DDA being the fastest. Figure 2(c) presents the modularity obtained by all the algorithms, showing that our DDA and DDA2 are the highest among all the algorithms. It should be remembered that the original DDA was modified to improve modularity, not for acceleration of the CNM[2]. Figure 2(d) details the results obtained by all the algorithms in a network of 1 million nodes and 5 million edges. Though slower than DDA (still 53 times faster than the former CNM), DDA2 provides the best modularity among all the algorithms (13% improvement over the former CNM), and is recommended for networks up to 1 million nodes.

4. CONCLUSIONS

We presented a new implementation and a modification of the DDA to speed it up in our framework, obtaining an improvement of 192 times faster than the original CNM. Large-scale networks up to 10 million nodes and 50 million edges were employed, showing that our modifications provide better modularity and require less time compared to CNM and its modifications. As a result, our improvements make it applicable to large-scale networks.

5. REFERENCES

[1] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. PRE, 70:066111, 2004.

[2] L. Danon, A. Diaz-Guilera, and A. Arenas. Effect of size heterogeneity on community identification in complex networks. *J. Stat. Mech.*, P11010, 2006.

[3] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks. arXiv:cs/0702048, 2007.

[4] K. Yuta, N. Ono, and Y. Fujiwara. A gap in the community-size distribution of a large-scale social networking site. arXiv:physics/0701168v2, 2007.