InvestorCliques (796315) – EU-Project Milestone 3.2

Parallel comunity detection*

Le Viet Hung†

Department of Computing Sciences,

Tampere University, Tampere Finland

* I thank Mr. Kestutis Baltakys and Prof Juho Kanniainen for comments and support. A joint paper on the same topic will be published later.

† http://www.investorcliques.eu.; hung.le@tuni.fi

SUMMARY

In the milestone 2.2, we have shown that clique percolation return a good accuracy but encounter computational bottleneck. This is a typical case of algorithms which aim at optimal result since community detection is classic NP-problem. Develop a parallel version of those algorithms is one of the solutions.

The purpose of this milestone is to implement the parallel version of the clique percolation methods. The challenge lies at the first step that enumerates all cliques of the networks. In addition, finding the optimal parameters also take significant time cause clique graph and it's connected components need to be recalculated multiple of times.

In addition, we also implemented greedy clique expansion (GCE) Lee *et al.* [1]. The idea behind clique based community detection is that community usually contain cliques. Similar to clique percolation, GCE encounters problem of computing time.

Parallel clique percolation

First of all, all maximal cliques of the networks need to be detected. For this purpose, we implemented the parallel version of the algorithm of Eppstein and Strash [2]. Basically, the degeneracy order of the graph is computed using the algorithm of Batagelj and Zaversnik [3] and the existing package (see the reference). Then the pivot Bron-Kerbosch algorithm [4] with this initial vertex ordering are performed in parallel. The routine and source code are documented in milestone 2.1 of this project.

Optimal parameter selection: for each minimum clique size, we perform the routine percolation method (milestone 2.2) in parallel as following: clique graph is created with nodes are cliques, links are the overlap between them, and the connected components of clique graph are the communities. Then the modularity of the detected communities is computed using the algorithm of Newman [5]. The one with the highest is selected.

Parallel greedy clique expansion

The sequential greedy clique expansion (GCE) is introduced by Lee *et al.* [1] as the following steps: i) clique enumeration ii) clique expansion for each detected clique and iii) duplicate removal. Compare with clique percolation, GCE is significant slower due to expansion routine which greedily finds and adds nodes to the existing seeds (cliques) to optimize the scoring function. In addition, choosing the optimal input parameter for the scoring function is another problem. It requires to repeat step ii) and iii) for multiple of

times. The accuracy of GCE depends on the parameters. For the investor network, clique percolation is more stable and return higher accuracy.

To improve it, we develop the parallel version. The advantage is that the algorithm can be paralleled. The first step is similar to clique percolation. We apply the parallel clique enumeration Eppstein and Strash [2]. For each clique, the greedy expansion routine is also performed in parallel.

A more detailed technical Python implementation of the algorithm is presented at: http://www.investorcliques.eu/category/programming/

DATA

We first use the input correlation network from our lab's previous work [6]. The data used in this study is the central register of shareholdings for Finnish stocks from Finnish central depository, provided by Euroclear Finland. Our sample data consists of the market-place transactions of 100 Finnish stocks consisting of investor's transactions around dot-com bubble from 1 January 1998 to 1 January 2002. A more detailed description of the data set is provided in Refs [6–10].

AVAILABILITY

Source code Python can be found at: http://www.investorcliques.eu/category/programming/

REFERENCE

https://gist.github.com/abhin4v/8304062

https://networkx.github.io/documentation/networkx-1.10/$_m$odules/networkx/algorithms/community

https://networkx.github.io/documentation/latest/$_m$odules/networkx/algorithms/community/modula

[1] C. Lee, F. Reid, A. McDaid, and N. Hurley, arXiv preprint arXiv:1002.1827 (2010).

[2] D. Eppstein and D. Strash, in *International Symposium on Experimental Algorithms* (Springer, 2011) pp. 364–375.

[3] V. Batagelj and M. Zaversnik, arXiv preprint cs/0310049 (2003).

[4] C. Bron and J. Kerbosch, Communications of the ACM **16**, 575 (1973).

[5] M. E. Newman, Proceedings of the national academy of sciences **103**, 8577 (2006).

[6] S. Ranganathan, M. Kivelä, and J. Kanniainen, PloS one **13**, e0198807 (2018).

[7] M. Grinblatt and M. Keloharju, Journal of financial economics **55**, 43 (2000).

[8] M. Tumminello, F. Lillo, J. Piilo, and R. N. Mantegna, New Journal of Physics **14**, 013041 (2012).

[9] K. Baltakys, J. Kanniainen, and F. Emmert-Streib, Scientific reports **8**, 8198 (2018).

[10] F. Musciotto, L. Marotta, J. Piilo, and R. N. Mantegna, Palgrave Communications **4**, 92 (2018).