

# Software Modules Clustering Using Social Network Algorithms

Nida Shahab<sup>1</sup>, Naveed Ahmad<sup>2</sup>, Maryam Dobarjeh<sup>1</sup>

<sup>1</sup>Knowledge Engineering and Discovery Research Innovation (KEDRI), School of Engineering Computer and Mathematical Sciences, Auckland University of Technology AUT, Auckland, 1010, New Zealand. Email: [nida.shahab@autuni.ac.nz](mailto:nida.shahab@autuni.ac.nz), [mgholami@aut.ac.nz](mailto:mgholami@aut.ac.nz)

<sup>2</sup>Department of Software Engineering, National University of Computer and Emerging Sciences FAST, Islamabad, 44000, Country. Email: [naveed.ahmad@nu.edu.pk](mailto:naveed.ahmad@nu.edu.pk)

## Summary

Software development life cycle continues even after deployment, and changes or enhancements made after deployment are the most complex. Especially, in the absence of design documentation, implementing these changes can be tricky as it can adversely affect the design thereby disturbing software's modular structure. So, it is imperative to quickly analyse and understand the modular structure of the software. Software Module Clustering Problem (SMCP) is one possible method to understand structure of complex software systems and enhance it without degrading the modular structure and violating the design rules. Various Artificial Intelligence (AI) techniques have been applied in the past providing an optimal modular structure, but they are time consuming. This research models SMCP as a community detection problem inspired by social networks. The results indicate that using community detection algorithm (CDA) an optimal solution can be achieved in terms of time and produces near optimal results for modularization quality.

## Background

Software Module Clustering Problems (SMCPs) involve partitioning software modules, represented as Module Dependency Graphs (MDGs) and grouping interdependent modules into clusters or subsystems to maximize cohesion and minimize coupling [1-3]. Such grouping is considered scalable, manageable and can be extended overtime [4]. During its life software evolves as the needs of stakeholders change [2,4] and due to frequent changes especially after deployment this evolution degrades the modular structure of software [5,6]. Therefore, when implementing these changes, it is important to quickly analyze and understand the high-level structure of the software which becomes a more daunting task in the absence of design documents [3]. If these changes are applied without considering the high-level structure it will deteriorate over time. SMCP can be deployed to modularize any software, given the dependencies between modules, to understand structure of software systems when applying these changes.

As aforementioned, MDG is used to illustrate software structure, where nodes represent modules and edges represent relationships between modules. SMCP is considered as a graph partitioning problem, which is an NP-hard problem [7]. To get optimal results various AI-Search Based Software Engineering (SBSE) heuristics guided by objective "Modularization Quality" MQ fitness function [1,8,9]. MQ provides balance between intra-edges and inter-edges by

minimizing inter clusters connections and maximizing intra-edges. A good module structure is regarded as one that has a high degree of cohesion and a low degree of coupling for MQ details refer [10].

The search-based techniques reported better results in terms of modular structure of software systems, but they are very time-consuming [11], which can be problematic when changes in software systems are applied immediately after change request. Moreover, SBSE techniques used for SMCPs are time consuming for larger graphs as their execution time depends on size of MDG. This research aims to develop a solution to SMCP which optimizes modularity i.e. near optimal software structure according to the design rules and is also time effective.

This research defines SMCP as community detection problem inspired by social networks, as by nature both these problems are quite similar i.e. structure network/software into communities/clusters of densely connected nodes, with the nodes belonging to different communities being only sparsely connected [12]. Community Detection Algorithms (CDA) have been used in various social and biological networks, provides a subset of vertices within which vertex to vertex connections are dense and between them the connections are less dense [13]. The effectiveness of community structure is often measured with quality metric called “modularity” ( $Q$ ), for details of  $Q$  refer [12]. Our experimental results show that CDA outperforms in solving SMCPs in terms of time.

## Methodology

We have applied CDA on standard dataset [1,9,14] in SBSE and compared our results in terms of time in seconds and MQ. The dataset is represented as MDGs and are un-weighted and directional graphs. Experiments were carried out on MATLAB and Pajek using Intel-Core-i3. CDAs were applied to find well modularized software structures in a reasonable amount of time using modularity  $Q$  [12] then for each clustered graph MQ is computed. The selected CDAs are [12,15,16,17] presented in Table. 1. All these algorithms are selected because they search for non-overlapping communities (grouping nodes into disjoint communities/clusters) this best fits the aim of SMCP since they are also group modules into disjoint clusters as shown in Fig. 1.

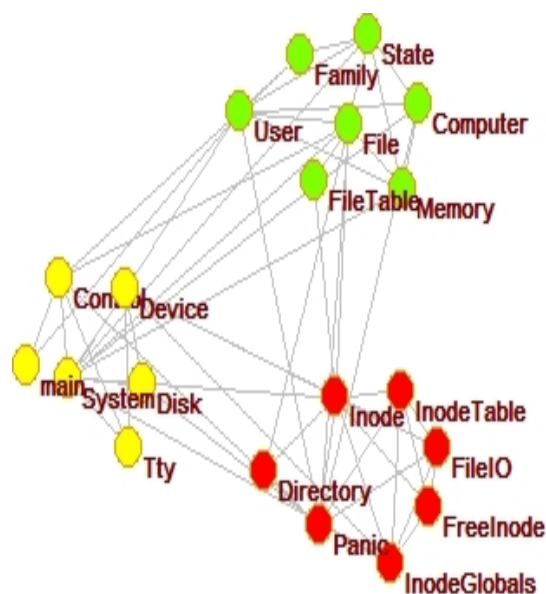


Fig. 1. Clustered modules of mtunis

## Results and Observations

The above-mentioned techniques are applied on SMCPs results based on computation time and MQ are presented in Table. 1. These findings are compared with some time efficient techniques [1,11].

Table. 1. Comparison of computation time and MQ found by CDAs and SBSE techniques

MDG	CDA								SBSE			
	Louvain-Method		Tabu-Search		Extremal-Optimization		Fast-Algorithm		MAEA		Tree Algo	
	Time-Sec	MQ	Time-Sec	MQ	Time-Sec	MQ	Time-Sec	MQ	Time-Sec	MQ	Time-Sec	MQ
mtunis	<b>0.00</b>	<b>2.23</b>	0.5	2.1	0.4	1.8	0.06	2.04	9.6	2.31	7.61	2.3
ispell	<b>0.00</b>	<b>2.31</b>	0.4	2.1	0.4	2.0	0.09	1.97	14.5	2.35	20.72	2.1
rsc	<b>0.00</b>	<b>1.92</b>	0.8	1.8	0.7	1.8	0.04	1.54	22.0	2.22	70.34	2.1
bison	<b>0.00</b>	<b>2.04</b>	0.8	2.1	1.1	1.6	0.20	1.7	70.6	2.65	87.21	2.5
grappa	<b>0.00</b>	<b>10.7</b>	2.2	10.8	2.4	6.7	0.04	6.63	591.3	12.5	---	---
incl	<b>0.00</b>	<b>12.1</b>	7.3	9.4	4.1	6.5	0.13	4.8	1001.3	13.5	---	---

From the above findings we can conclude that CDAs are extremely fast compared to SBSE techniques. Louvain method and Fast algorithm outperformed all the above community detection algorithms in terms of time, but Fast algorithm doesn't provide good software module clustering. In terms of software modularization, the Louvain method and Tabu search provide some reasonable MQ values. We can consider the Louvain method as the most reasonable technique for software modularization using social networks algorithms because it is time efficient and provides near optimal clustering.

## Conclusion

Existing solutions of SMCPs are observed to be time consuming for larger graphs. Since CDAs have proved to be fast in terms of computation time are applied in this research on SMCPs. The applied algorithms outperformed in terms of computation time on SMCPs even for larger graphs and provide near to optimal values in terms of modularity. Therefore, we can use them to understand high level structure of the system during evolution when quick changes are to be applied.

## Acknowledgement

The authors are grateful to Prof. Dr Mark Harman, University College London (UCL) for providing the dataset used in this research.

## References

- [1] J. Huang and J. Liu, 2016. 'A similarity-based modularization quality measure for software module clustering problems', *Inf Sci (N Y)*, vol. 342, pp. 96–110, doi: 10.1016/j.ins.2016.01.030.
- [2] K. M. Mark, H. Robert, and M. Hierons, 2003. 'A Multiple Hill Climbing Approach to Software Module Clustering', in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings*, Amsterdam, Netherlands: IEEE, pp. 315–324. doi: 10.1109/ICSM.2003.1235437.
- [3] D. Doval, S. Mancoridis, and B. S. Mitchell, 1999. 'Automatic Clustering of Software Systems using a Genetic Algorithm', in *STEP '99. Proceedings Ninth International Workshop Software Technology and Engineering Practice*, Pittsburgh, PA, USA: IEEE, p. 73. doi: 10.1109/STEP.1999.798481.

- [4] B. S. Mitchell, 2003. 'A Heuristic Approach to Solving the Software Clustering Problem', in International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings, Amsterdam, Netherlands: IEEE, pp. 285–288. doi: 10.1109/ICSM.2003.1235432.
- [5] A. C. Kumari and K. Srinivas, 2016. 'Hyper-heuristic approach for multi-objective software module clustering', Journal of Systems and Software, vol. 117, pp. 384–401, doi: 10.1016/j.jss.2016.04.007.
- [6] V. Cortellessa, R. Mirandola, and P. Potena, 2015. 'Managing the evolution of a software architecture at minimal cost under performance and reliability constraints', Sci Comput Program, vol. 98, no. P4, pp. 439–463, doi: 10.1016/j.scico.2014.06.001.
- [7] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner, 1998. 'Using Automatic Clustering to Produce High-Level System Organizations of Source Code', in Proceedings. 6th International Workshop on Program Comprehension. IWPC'98, Ischia, Italy: IEEE, pp. 45–52. doi: 10.1109/WPC.1998.693283.
- [8] I. Hussain, A. Khanum, A. Q. Abbasi, and M. Y. Javed, 2015. 'A Novel Approach for Software Architecture Recovery using Particle Swarm Optimization', The International Arab Journal of Information Technology, vol. 12, no. 1.
- [9] K. Jeet and R. Dhir, 2015. 'Software Architecture Recovery using Genetic Black Hole Algorithm', ACM SIGSOFT Software Engineering Notes, vol. 40, no. 1, pp. 1–5, doi: 10.1145/2693208.2693230.
- [10] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner, 1999. 'Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures', in Proceedings IEEE International Conference on Software Maintenance - (ICSM'99), Oxford, UK: IEEE, pp. 50–50. doi: 10.1109/ICSM.1999.792498.
- [11] S. Mohammadi and H. Izadkhah, 2019. 'A new algorithm for software clustering considering the knowledge of dependency between artifacts in the source code', Inf Softw Technol, vol. 105, pp. 252–256, doi: 10.1016/j.infsof.2018.09.001.
- [12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, 2008. 'Fast unfolding of communities in large networks', doi: 10.1088/1742-5468/2008/10/P10008.
- [13] M. Girvan and M. E. J. Newman, 2001. 'Community structure in social and biological networks', Proceedings of the National Academy of Sciences, doi: 10.1073/pnas.122653799.
- [14] K. Praditwong, M. Harman, and X. Yao, 2011. 'Software module clustering as a multi-objective search problem', IEEE Transactions on Software Engineering, vol. 37, no. 2, pp. 264–282, doi: 10.1109/TSE.2010.26.
- [15] A. Arenas, A. Fernández, and S. Gómez, 2008. 'Analysis of the structure of complex networks at different resolution levels', New J Phys, vol. 10, doi: 10.1088/1367-2630/10/5/053039.
- [16] J. Duch and A. Arenas, 2005. 'Community detection in complex networks using Extremal Optimization', American Physical Society, vol. 72, no. 2, doi: 10.1103/PhysRevE.72.027104.
- [17] M. E. J. Newman, 2003. 'Fast algorithm for detecting community structure in networks', American Physical Society, vol. 69, no. 6, doi: 10.1103/PhysRevE.69.066133.