Local Motif Clustering via (Hyper)Graph Partitioning

Adil Chhabra[1], **Marcelo Fonseca Faraj**[2], **Christian Schulz**[3]

Heidelberg University, Germany
[1]adil.chhabra@stud.uni-heidelberg.de [2]marcelofaraj@informatik.uni-heidelberg.de
[3]christian.schulz@informatik.uni-heidelberg.de

Abstract

Local clustering consists of finding a good cluster around a seed node in a graph. Recently local motif clustering has been proposed: it is a local clustering approach based on motifs rather than edges. Since this approach is recent, most algorithms to solve it are extensions of statistical and numerical methods previously used for local clustering, while combinatorial approaches are still few and simple. In this work, we build a (hyper)graph to represent the motif-distribution around the seed node. We solve this model using sophisticated (hyper)graph partitioners. On average, our algorithm computes clusters six times faster and three times better than the state-of-the-art for the triangle motif.

Introduction

Given a graph and a seed node, the local clustering problem consists of identifying a *well-characterized* cluster which contains the seed node. The quality of a cluster can be measured by its *conductance* (Kannan, Vempala, and Vetta 2004), which is NP-hard (Wagner and Wagner 1993) to optimize. Ideally memory and space complexity of local clustering heuristics dependent on cluster size rather than graph size. In traditional clustering, a cluster is evaluated by its edge distribution. A promising novel approach named *local motif clustering* evaluates clusters based on motif distribution. Since this approach is recent, most algorithms for it are either extensions of statistical and numerical methods previously proposed for local clustering or simple combinatorial methods (Yin et al. 2017; Zhang et al. 2019; Meng et al. 2019; Murali, Potika, and Pollett 2020). We propose a combinatorial algorithm to solve the local motif clustering problem based on sophisticated (hyper)graph partitioning algorithms. On average, our algorithm computes clusters six times faster and three times better than the state-of-the-art.

Related Work

Rohe and Qin (2013) grow a cluster initialized with the seed node by adding nodes from cut triangles. Zhang et al. (2019) obtain a cluster using a simplified spectral approach based on approximate local motif spectra computed via random walk. Meng et al. (2019) compute a cluster by optimizing a fuzzy adaptation of the modularity metric. Zhou et al. (2021) find a cluster by approximating the motif distribution vector via random walk and applying vector-based partitioning (Spielman and Teng 2013). Shang et al. (2022) grow a cluster initialized with the seed node by adding neighbors based on motif degree to optimize for motif-based modularity. Yin et al. (2017) propose MAPPR, an algorithm based on PageRank which optimizes for *motif conductance* (ϕ_μ) (Benson, Gleich, and Leskovec 2016).

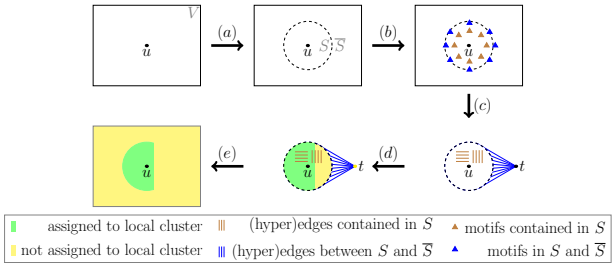
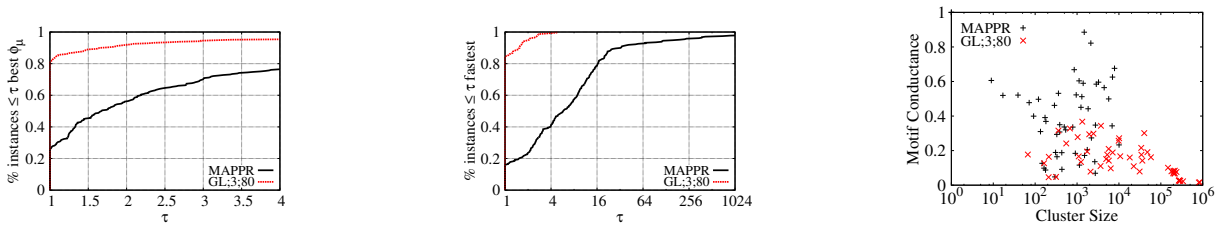


Figure 1: (a) Get ball S. (b) Enumerate motifs. (c) Build H_μ. (d) Partition H_μ. (e) Convert partition to cluster around u.

Local Motif Graph Clustering

Overall Strategy. For a graph G, a seed node $u \in V(G)$, and a motif μ, our algorithm has 4 phases. First, select a set $S \subseteq V$ containing u and close-by nodes (a *ball around* u). Second, enumerate the set M of occurrences of μ containing nodes in S. Third, build a (hyper)graph model H_μ such that the motif conductance in G is computable from H_μ. Fourth, partition H_μ using a high-quality (hyper)graph partitioner such that the obtained partition directly converts back to G as a cluster around u. Fig. 1 shows the four phases of our algorithm. We pick the best motif conductance out of α repetitions of the overall strategy with different balls around u and β repetitions of the partitioning phase with random imbalance constraints (Alg. 1). For the graph-based version of H_μ, we run a special label propagation on each obtained partition to try to reach a local minimum motif conductance.

Ball around Seed Node, Motif Enumeration. We select S via a fixed-depth breadth-first search (BFS) rooted on u, i.e., by computing ℓ layers of this BFS and including touched nodes in S. For α repetitions of this phase, we use different amounts ℓ of layers for a better exploration. This is done in time linear on the subgraph induced by S



(a) Motif conductance perf. profile. (b) Running time perf. profile. (c) Found clusters for com-orkut.

Figure 2: Comparison of the graph–based version of our algorithm against the state-of-the-art.

Algorithm 1: Local Motif Graph Clustering

```
for i = 1, ..., α do
    S ← ball around u; M ← Enumerate motifs in S
    Build (hyper)graph model H_μ based on S and M
    for j = 1, ..., β do
        Partition model H_μ into (C, C̄), where u ∈ C
        if C* = ∅ ∨ φ_μ(C) < φ_μ(C*) then C* ← C
Convert C* into a local motif cluster in G
```

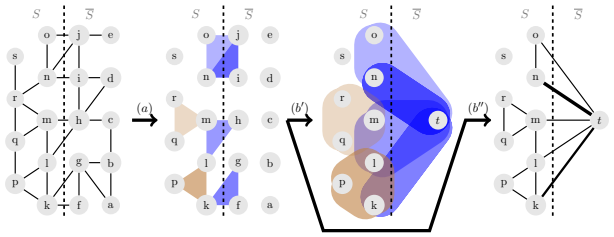


Figure 3: Example construction of H_μ. (a) Enumerate motifs touching S. (b') Build hypergraph H_μ. (b") Build graph H_μ.

and its neighborhood. In this work, our focus is the triangle motif. We enumerate triangles with a polynomial algorithm proposed in (Chiba and Nishizeki 1985). We apply this algorithm on the subgraph induced by S and its neighborhood.

(Hyper)Graph Model. Our algorithm has a configuration using a hypergraph model for partitioning and another using a graph model for partitioning. Our hypergraph model is $H_\mu = (S \cup \{t\}, \mathcal{E})$ with unweighted nodes and a set $\mathcal{E}$ of nets with a net e for each motif $G' = (V', E') \in M$ such that $e = V'$ if $V' \subseteq S$, and $e = V' \cap S \cup \{t\}$ otherwise. In the former case the net has weight one, in the latter case its weight equals the number of motif occurrences in M represented by it. See Fig. 1(c), Fig. 3(b'), and Theo. 1. Our graph model is $H_\mu = (S \cup \{t\}, E_\mu)$ with unweighted nodes where E_μ has an edge e for each pair of nodes sharing a motif $G' = (V', E') \in M$ provided that at least one of its endpoints is contained in S. The weight of an edge equals the number of motif occurrences containing both its endpoints. See Fig. 1(c), Fig. 3(b"), and Theo. 1.

Theorem 1. *A cluster C in H_μ, $t \notin C$, implies a cluster in G whose motif conductance is the ratio of its cut-net (edge-cut) to its volume in H_μ, assumed Volume$_\mu(S) \leq$ Volume$_\mu(\overline{S})$.*

Partitioning. For partitioning our hypergraph and graph models, we respectively use KaHyPar (Schlag et al. 2016) and KaHIP (Sanders and Schulz 2022). For both versions of our algorithm, we look for a partition where u and t are in different blocks to ensure consistency. We force it by assigning u to the block that does not contain t after the partition is computed (before computing ϕ_μ).

Graph	GL;3;80			MAPPR		
	ϕ_μ	$\lvert C\rvert$	t(s)	ϕ_μ	$\lvert C\rvert$	t(s)
com-amazon	**0.037**	64	0.22	0.153	58	2.68
com-dblp	**0.115**	56	0.38	0.289	35	3.04
com-youtube	**0.172**	1443	7.93	0.910	2	10.44
com-livejournal	**0.244**	387	8.17	0.507	61	173.80
com-orkut	**0.150**	13168	496.94	0.407	511	923.26
com-friendster	**0.368**	10610	1339.99	0.741	121	16565.99
Overall	**0.181**	823	12.67	0.500	50	79.34

Table 1: Average comparison against state-of-the-art.

Experimental Evaluation

Methodology and Instances. We implemented our algorithm in KaHIP (Sanders and Schulz 2011) and KaHyPar (Schlag et al. 2016). We used a 64-core AMD EPYC 7702P Linux server with 1 TB RAM. Our test set is the same used in MAPPR paper (Yin et al. 2017). For each graph, we use 50 random seed nodes. We compare results against the globally best clusters found by MAPPR (Yin et al. 2017) with standard parameters. Our algorithm uses parameters: graph-based, $\alpha = 3$, $\beta = 80$, label propagation.

Comparison against State-of-the-Art Fig. 2a and 2b show performance profiles. Our algorithm gets the best or equal ϕ_μ for 80% of the instances, MAPPR gets the best or equal ϕ_μ for 25% of the instances. This is due to the fact that by performing multiple cluster constructions and refinements our algorithm can explore the solution space better than MAPPR which simply uses the APPR algorithm. Our algorithm is faster than MAPPR for 84% of the instances, besides being on average a factor 6.3 faster. Table 1 shows average results for each graph and average results overall. Our algorithm outperforms MAPPR with respect to ϕ_μ and running time for every graph and overall. Figure 2c plots ϕ_μ vs cluster size for the com-orkut graph. Clusters found by our algorithm are localized in the lowest half of the chart, while the clusters found by MAPPR are widespread.

Conclusion

We proposed an algorithm which computes local motif clustering via partitioning of (hyper)graph models. Our algorithm computes clusters that have on average one third of the motif conductance value than MAPPR while being 6.3 times faster on average and removing the necessity of a preprocessing motif-enumeration on the whole network.

References

Benson, A. R.; Gleich, D. F.; and Leskovec, J. 2016. Higher-order organization of complex networks. *Science*, 353(6295): 163–166.

Chiba, N.; and Nishizeki, T. 1985. Arboricity and subgraph listing algorithms. *SIAM J. Comp.*, 14(1): 210–223.

Kannan, R.; Vempala, S.; and Vetta, A. 2004. On clusterings: Good, bad and spectral. *JACM*, 51(3): 497–515.

Meng, T.; Cai, L.; He, T.; Chen, L.; and Deng, Z. 2019. Local higher-order community detection based on fuzzy membership functions. *IEEE Access*, 7: 128510–128525.

Murali, M.; Potika, K.; and Pollett, C. 2020. Online local communities with motifs. In *2020 Second Intl. Conf. on Transdisciplinary AI (TransAI)*, 59–66. Los Alamitos, CA, USA: IEEE Computer Society.

Rohe, K.; and Qin, T. 2013. The blessing of transitivity in sparse and stochastic networks. *arXiv preprint arXiv:1307.2302*.

Sanders, P.; and Schulz, C. 2011. Engineering Multilevel Graph Partitioning Algorithms. In *Proc. of the 19th European Symp. on Algorithms*, volume 6942 of *LNCS*, 469–480. Springer.

Sanders, P.; and Schulz, C. 2022. KaHIP – Karlsruhe High Qualtity Partitioning Homepage. http://algo2.iti.kit.edu/documents/kahip/index.html. Accessed: 2022-05-31.

Schlag, S.; Henne, V.; Heuer, T.; Meyerhenke, H.; Sanders, P.; and Schulz, C. 2016. k-way Hypergraph Partitioning via n-Level Recursive Bisection. In *ALENEX*, 53–67.

Shang, R.; Zhang, W.; Zhang, J.; Feng, J.; and Jiao, L. 2022. Local community detection based on higher-order structure and edge information. *Physica A: Statistical Mechanics and its Applications*, 587: 126513.

Spielman, D. A.; and Teng, S.-H. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comp.*, 42(1): 1–26.

Wagner, D.; and Wagner, F. 1993. Between Min Cut and Graph Bisection. In *FOCS*, 744–750. Springer.

Yin, H.; Benson, A. R.; Leskovec, J.; and Gleich, D. F. 2017. Local higher-order graph clustering. In *23rd ACM SIGKDD*, 555–564.

Zhang, Y.; Wu, B.; Liu, Y.; and Lv, J. 2019. Local community detection based on network motifs. *Tsinghua Science and Technology*, 24(6): 716–727.

Zhou, D.; Zhang, S.; Yildirim, M. Y.; Alcorn, S.; Tong, H.; Davulcu, H.; and He, J. 2021. High-order structure exploration on massive graphs: A local graph clustering perspective. *ACM TKDD*, 15(2): 1–26.