Faster Local Motif Clustering via Maximum Flows

Adil Chhabra
Heidelberg University, Germany

Marcelo Fonseca Faraj
Heidelberg University, Germany

Christian Schulz
Heidelberg University, Germany

Abstract

Local clustering aims to identify a cluster within a given graph that includes a designated seed node or a significant portion of a group of seed nodes. This cluster should be well-characterized, i.e., it has a high number of internal edges and a low number of external edges. In this work, we propose `SOCIAL`, a novel algorithm for local motif clustering which optimizes for motif conductance based on a local hypergraph model representation of the problem and an adapted version of the max-flow quotient-cut improvement algorithm (`MQI`). In our experiments with the triangle motif, `SOCIAL` produces local clusters with an average motif conductance 1.7% lower than the state-of-the-art, while being up to multiple orders of magnitude faster.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases local motif clustering, motif conductance, maximum flows, max-flow quotient-cut improvement

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.34

Related Version *Full Version*: `https://arxiv.org/pdf/2301.07145.pdf`

Supplementary Material *Software (Code)*:
`https://github.com/LocalClustering/HeidelbergMotifClustering`

Funding DFG grant SCHU 2567/5-1.

1 Introduction

Graphs are a fundamental tool for representing complex systems and relationships in a wide range of contexts. They can be used to model everything from data dependencies and social networks to web links and email interactions. With the massive expansion of data in recent years, many real-world graphs have grown to enormous sizes, making it challenging to analyze them. In particular, many applications only require analyzing a small, localized portion of a graph rather than the entire graph, which is the case for community-detection on Web [12] and social [21] networks as well as structure-discovery in bioinformatics [47] networks, among others. Those real-world applications are usually preceded by or modeled as a *local clustering* problem. Local clustering aims at identifying a specific cluster within a given graph that includes a designated seed node or a portion of a group of seed nodes, and is *well-characterized*, i.e., it consists of many internal edges and few external edges. More specifically, the quality of a community can be quantified by specific metrics such as *conductance* [22]. Since minimizing conductance is NP-hard [48], approximate and heuristic approaches are used in practice. Given the nature and scale of the problem, these approaches should ideally require time and memory dependent only on the size of the found cluster.

The local clustering problem has been investigated both theoretically [1] and experimentally [29], and has been solved using a wide variety of techniques, including statistical [9, 24], numerical [30, 32], and combinatorial [35, 15] methods. While traditional approaches to local

clustering typically consider the edge distribution when evaluating the quality of a local community, novel methods [49, 50, 33, 34, 7] have shifted focus to finding local communities based on the distribution of *motifs*, higher-order structures within the graph. These works provide empirical evidence that this approach, which can be called *local motif clustering*, is effective at detecting high-quality local communities. Nevertheless, since this local clustering perspective is relatively new, there are still many opportunities to improve upon current approaches and discover more efficient algorithms for finding high-quality solutions.

Contribution. In this work, we propose a novel algorithm for local motif clustering which optimizes for motif conductance by combining the strongly local hypergraph model from Chhabra et al. [7] with an adapted version of the fast and effective algorithm *max-flow quotient-cut improvement* (`MQI`) [26]. Our algorithm `SOCIAL`, which stands for *faSter mOtif Clustering vIa mAximum fLows*, starts by building a hypergraph model which is an exact representation for the motif-distribution around the seed node on the original graph [7]. Using this model, we create a flow model in which certain cuts correspond one-to-one with sub-sets of the initial cluster that include the seed node and have lower motif conductance than that of the whole cluster. We then use a push-relabel algorithm to either find such a cut and repeat the process recursively, or to prove that the current cluster is optimal among all its sub-clusters containing the seed node. In our experiments with the triangle motif, `SOCIAL` produces communities with a motif conductance value that is 1.7% lower than the state-of-the-art on average, while also being up to multiple orders of magnitude faster.

2 Preliminaries

Graphs. Let $G = (V = \{0, \ldots, n-1\}, E)$ be an *undirected graph* with no multiple or self edges allowed, such that $n = |V|$ and $m = |E|$. Let $c : V \to \mathbb{R}_{\geq 0}$ be a node-weight function, and let $\omega : E \to \mathbb{R}_{>0}$ be an edge-weight function. We generalize c and ω functions to sets, such that $c(V') = \sum_{v \in V'} c(v)$ and $\omega(E') = \sum_{e \in E'} \omega(e)$. Let $N(v) = \{u : \{v, u\} \in E\}$ be the *open neighborhood* of v, and let $N[v] = N(v) \cup \{v\}$ be the *closed neighborhood* of v. We generalize the notations $N(.)$ and $N[.]$ to sets, such that $N(V') = \cup_{v \in V'} N(v)$ and $N[V'] = \cup_{v \in V'} N[v]$. A graph $G' = (V', E')$ is said to be a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. When $E' = E \cap (V' \times V')$, G' is the subgraph *induced* in G by V'. Let $\overline{V'} = V \setminus V'$ be the *complement* of a set $V' \subseteq V$ of nodes. Let a *motif* μ be a connected graph. *Enumerating* the motifs μ in a graph G consists of building the collection M of all occurrences of μ as a subgraph of G. Let $d(v)$ be the *degree* of node v and Δ be the maximum degree of G. Let $d_\omega(v)$ be the *weighted degree* of a node v and Δ_ω be the maximum weighted degree of G. Let $d_\mu(v)$ be the *motif degree* of a node v, i.e., the number of motifs $\mu \in M$ which contain v. We generalize the notations $d(.)$, $d_\omega(.)$, and $d_\mu(.)$ to sets, such that the *volume* of V' is $d(V') = \sum_{v \in V'} d(v)$, the *weighted volume* of V' is $d_\omega(V') = \sum_{v \in V'} d_\omega(v)$, and the *motif volume* of V' is $d_\mu(V') = \sum_{v \in V'} d_\mu(v)$. Let a *spanning forest* of G be an acyclic subgraph of G containing all its nodes. Let the *arboricity* of G be the minimum amount of spanning forests of G necessary to cover all its edges.

Local Motif Clustering. In the *local graph clustering* problem, a graph $G = (V, E)$ and a seed node $u \in V$ are taken as input and the goal is to detect a *well-characterized cluster* (or *community*) $C \subset V$ containing u. A high-quality cluster C usually contains nodes that are densely connected to one another and sparsely connected to $\overline{C}$. There are many functions to quantify the quality of a cluster, such as *modularity* [5] and *conductance* [22].

The conductance metric is defined as $\phi(C) = |E'|/\min(d(C), d(\overline{C}))$, where $E' = E \cap (C \times \overline{C})$ is the set of edges shared by a cluster C and its complement. *Local motif graph clustering* is a generalization of local graph clustering where a motif μ is taken as an additional input and the computed cluster optimizes a clustering metric based on μ. In particular, the *motif conductance* $\phi_\mu(C)$ of a cluster C is defined by Benson et al. [4] as a generalization of the conductance in the following way: $\phi_\mu(C) = |M'|/min(d_\mu(C), d_\mu(\overline{C}))$, where M' are all the motifs μ which contain at least one node in C and one node in $\overline{C}$. Note that, if the motif under consideration is simply an edge, then $|M'|$ is the edge-cut and $\phi_\mu(C) = \phi(C)$.

Hypergraphs. Let $H = (\mathcal{V} = \{0, \ldots, \mathfrak{n}-1\}, \mathcal{E})$ be an *undirected hypergraph* with no multiple or self hyperedges allowed, with $\mathfrak{n} = |\mathcal{V}|$ *nodes* and $\mathfrak{m} = |\mathcal{E}|$ *hyperedges* (or *nets*). A net is defined as a subset of $\mathcal{V}$. The nodes that compose a net are called *pins*. Let $c : \mathcal{V} \to \mathbb{R}_{\geq 0}$ be a node-weight function, and let $\omega : \mathcal{E} \to \mathbb{R}_{>0}$ be a net-weight function. We generalize c and ω functions to sets, such that $c(\mathcal{V}') = \sum_{v \in \mathcal{V}'} c(v)$ and $\omega(\mathcal{E}') = \sum_{e \in \mathcal{E}'} \omega(e)$. A node $v \in \mathcal{V}$ is *incident* to a net $e \in \mathcal{E}$ if $v \in e$. Let $\mathcal{I}(v)$ be the set of incident nets of v, let $d(v) := |\mathcal{I}(v)|$ be the *degree* of v, and let $d_\omega(v) := \omega(\mathcal{I}(v))$ be the *weighted degree* of v. We generalize the notations $d(.)$ and $d_\omega(.)$ to sets, such that the *volume* of $\mathcal{V}'$ is $d(\mathcal{V}') = \sum_{v \in \mathcal{V}'} d(v)$ and the *weighted volume* of $\mathcal{V}'$ is $d_\omega(\mathcal{V}') = \sum_{v \in \mathcal{V}'} d_\omega(v)$. Two nodes are *adjacent* if they are incident to the same net. Let the number of pins $|e|$ in a net e be the *size* of e. We define the *contraction* operator as $/$ such that $H/\mathcal{V}'$, with $\mathcal{V}' \subseteq \mathcal{V}$, is the hypergraph obtained by contracting the nodes from $\mathcal{V}'$ of H. This contraction consists of substituting all the nodes in $\mathcal{V}'$ by a single representative node x, removing nets totally contained in $\mathcal{V}'$, and substituting all the pins in $\mathcal{V}'$ by a single pin x in each of the remaining nets. Given a cluster $\mathcal{V}' \subseteq \mathcal{V}$, the *cut* or *cut-net* $cut(\mathcal{V}')$ of $\mathcal{V}'$ consists of the total weight of the nets crossing the cluster, i.e., $cut(\mathcal{V}') = \sum_{e \in \mathcal{E}'} \omega(\mathcal{E}')$, in which $\mathcal{E}' := \left\{e \in \mathcal{E} : e \cap \mathcal{V}' \neq \emptyset, e \cap \overline{\mathcal{V}'} \neq \emptyset\right\}$.

Flows. Let $\mathcal{N} = (V, E)$ be a directed flow network. A directed flow network has one source node $s \in V$, one sink node $t \in V$, and a set of remaining nodes $V \setminus \{s, t\}$. All edges $e = (u, v)$ in a directed flow network are directed and associated with a nonnegative capacity $cap(u, v)$. An s-t flow is a function $f : V \times V \to \mathbb{R}_{>0}$ which satisfies a *capacity* constraint, i. e., $f(u, v) \leq cap(u, v)$, a *symmetry* constraint, i. e., $\forall u, v \in V : f(u, v) = -f(v, u)$, and a *flow conservation* constraint, i. e., $\forall u \in V \setminus \{s, t\} : \sum_{v \in V} f(u, v) = 0$. An edge (u, v) is called *saturated* if $cap(u, v) = f(u, v)$; The total amount of flow moved from s to t is defined as the *value* $|f|$ of f and is computed as follows: $|f| = \sum_{u \in V} f(u, t) = \sum_{v \in V} f(s, v)$. A given s-t flow f in $\mathcal{N}$ is *maximum* if, for any s-t flow f' in $\mathcal{N}$, $|f'| \leq |f|$. Let $\mathcal{N}_f = (V, E_f)$ be the *residual graph* associated with a given flow f on $\mathcal{N}$, such that $E_f = \{(u, v) \in V \times V : cap(u, v) - f(u, v) > 0\}$. According to the Max-Flow Min-Cut Theorem [13], the value $|f|$ of a maximum s-t flow f on $\mathcal{N}$ equals the weight of a minimum s-t cut on $\mathcal{N}$, i. e., a 2-way partition of $\mathcal{N}$ where edge weights equal edge capacities, s and t are in distinct blocks, and the total weight of the cut edges is minimum. To find the sink side of the minimum cut associated with a maximum flow in $\mathcal{N}$, a reverse breadth-first search can be performed on $\mathcal{N}$ starting at the sink node t.

Push-Relabel. For each node u in a directed flow network $\mathcal{N}$, let $d(u)$ be its potential and $exc(u) = \sum_{v \in V}(f(v, u) - f(u, v))$ be its *excess*. A node u is called *active* if $exc(u) > 0$. An edge (u, v) is called *admissible* if $cap(u, v) - f(u, v) > 0$ and $d(u) = d(v) + 1$. The push-relabel [16] algorithm builds a maximum flow by computing a succession of *preflows*, i. e., flows where the flow conservation constraint is relaxed and replaced by $\forall u \in V \setminus \{s, t\} : exc(u) \geq 0$. In the initial preflow, all out-edges of s are saturated, $\forall u \in V \setminus \{s\} : d(u) = 0$, and $d(s) = |V|$.

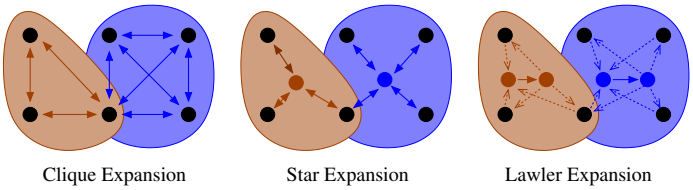


Figure 1 Net expansion techniques. Nodes and nets of the hypergraph are respectively represented by black circles and colored areas around them. Artificial nodes and edges are respectively represented by circles and arrows with same color as the corresponding net. Bidirectional arrows represent edges in both directions. Solid and dashed edges have finite and infinite weight, respectively.

The initial preflow is evolved via operations *push*, i. e., sending as much flow as possible from an active node through an admissible edge, and *relabel*, i. e., increasing the potential of a node until it becomes active. Preflows induce minimum sink-side cuts, so a maximum flow and a minimum cut are obtained once no node is active.

Flows on Hypergraphs. A common technique to solve flow problems on hypergraphs consists of transforming them in directed graphs and then applying traditional graph-based techniques on them. Among the existing transformations [46, 27], we highlight *clique expansion*, *star expansion*, and *Lawler expansion*. In the *clique expansion*, each net is represented by a clique, i.e., a set of edges connecting each pair of its pins in both directions. In this approach, the weight of each edge is equal to weight of the corresponding net e divided by $|e| - 1$ and parallel edges are substituted by a single edge whose weight is the sum of the weights of the removed edges. In the *star expansion*, each net is represented by an auxiliary artificial node connected to its pins by edges in both directions. In this expansion, the edges have the same weight as the corresponding net. In the *Lawler expansion*, each net e is represented by two auxiliary artificial nodes w_1 and w_1 and a collection of edges. In particular, there is a directed edge (w_1, w_2) which has the same weight as the corresponding net. Additionally, each pin of the corresponding net has an out-edge to w_1 and an in-edge from w_2, each of them with weight infinity. The three transformation approaches are exemplified in Figure 1.

2.1 Related Work

Motif-based clustering has been widely studied in the literature, with works such as [3, 49, 25, 36, 44] partitioning all the nodes of a graph into clusters based on motifs. We also address the topic of clustering based on motifs, but our focus is on identifying clusters in the immediate vicinity of a specific seed node, rather than on the entire graph. Several works [24, 30, 32, 11, 42] propose local clustering algorithms on graphs, but they do not focus on optimizing for motif-based metrics like our work. Instead, they use metrics based on edges, like conductance and modularity. Multiple works [45, 14, 20, 31] propose local clustering algorithms on hypergraphs. These algorithms are not designed for local graph clustering based on motifs, but for local hypergraph clustering. In one of them [45], the authors utilize a hypergraph extension of `FlowImprove` [2], which is itself an extension of the `MQI` [26] technique. Similarly, we also extend `MQI` to hypergraphs, then we use it as one of the steps of our algorithm `SOCIAL`. In this section, we review previous work on local graph clustering based on motifs, which is the focus of our work.

Rohe and Qin [38] propose a local clustering algorithm based on triangle motifs. Their algorithm starts by initializing a cluster containing only the seed node, and iteratively grows this cluster. Particularly, the algorithm greedily inserts nodes contained in at least a

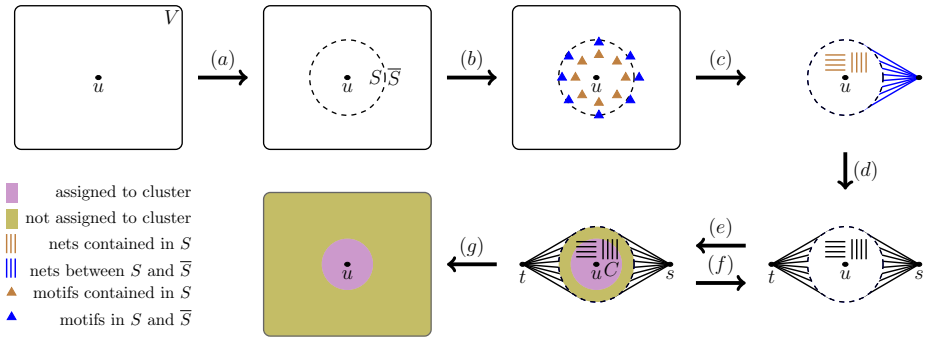


Figure 2 Illustration of the phases of SOCIAL. (a) Given a seed node u and a graph G, a ball S around u is selected. (b) Motif occurrences of μ with at least a node in S are enumerated. (c) The hypergraph model H_μ is built by converting motifs into nets and contracting $\overline{S}$ into a single node. The ball S is taken as the initial cluster C_0. (d) The flow model $\mathcal{N}$ is built based on C_0 in H_μ. (e) A cluster $C \subseteq C_0$ containing u is found using maximum flows. (f) While $C \subset C_0$, the model $\mathcal{N}$ is rebuilt based on C, which is taken as the initial cluster C_0. (g) When eventually $C = S$, C is converted in a local cluster around the seed node in G.

predefined amount of cut triangles. Huang et al. [19] recover local communities containing a seed node in online and dynamic setups based on higher-order graph structures named Trusses [10]. They define the k-truss of a graph as its largest subgraph whose edges are all contained in at least $(k-2)$ triangle motifs, hence trusses are a graph structure based on the frequency of triangles. The authors use indexes to search for k-truss communities in time proportional to the size of the recovered community.

Yin et al. [49] propose MAPPR, a local motif clustering algorithm based on the Approximate Personalized PageRank (APPR) method. In a preprocessing phase, MAPPR enumerates the motif of interest in the entire input graph and constructs a weighted graph W, in which edges only exist between nodes that appear in at least one instance of the motif, and their edge weight is equal to the number of occurrences of the motif containing these two endpoints. Afterward, MAPPR uses an adapted version of the APPR method to find local communities in the weighted graph constructed in the preprocessing phase. MAPPR is able to extract local communities from directed input graphs, something that cannot be done using APPR alone.

Zhang et al. [50] propose LCD-Motif, an algorithm that addresses the local motif clustering problem using a modified version of the spectral method. LCD-Motif has two main differences in comparison to the traditional spectral motif clustering method. First, instead of computing singular vectors, the algorithm performs random walks to identify potential members of the searched cluster. They use the span of a few dimensions of vectors, obtained through random walks, as an approximation for the local motif spectra. Second, instead of using k-means for clustering, LCD-Motif searches for the minimum 0-norm vector within the previously mentioned span, which must contain the seed nodes in its support vector.

Meng et al. [33] propose FuzLhocd, a local motif clustering algorithm that uses fuzzy arithmetic to optimize a modified version of modularity. Given a seed node, FuzLhocd starts by detecting probable core nodes of the targeted local community using fuzzy membership. After identifying the probable core nodes of the target local community using fuzzy membership, the algorithm expands these nodes using another fuzzy membership to form a cluster.

Zhou et al. [51] propose HOSPLOC, a local motif clustering algorithm that uses a motif-based random walk to compute a distribution vector, which is then truncated and used in a vector-based partitioning method. The algorithm begins by approximately estimating the

distribution vector through a motif-based random walk. To further refine the computation and focus on the local region, HOSPLOC sets all small vector entries to 0. After this preprocessing step, the algorithm applies a vector-based partitioning method [43] on the resulting distribution vector in order to identify a local cluster.

Shang et al. [41] propose HSEI, a local motif clustering algorithm that uses motif and edge information to grow a cluster from a seed node. The algorithm begins by creating an initial cluster consisting of only the seed node. It then adds nodes to the cluster from the seed's neighborhood, selecting them based on their motif degree. The cluster is expanded using a motif-based extension of the modularity function.

Chhabra et al. [7] propose an algorithm to solve the local motif clustering problem using powerful (hyper)graph partitioning tools [39, 40, 17, 18]. Their algorithm first uses a breadth-first search to select a ball containing the seed node and nearby nodes. Next, they enumerate motif occurrences within the ball and build a (hyper)graph model which allows them to compute the motif conductance of any cluster within the ball. They then partition their model into two blocks using a high-quality (hyper)graph partitioning algorithm, and refine the solution for motif conductance.

3 Local Motif Clustering via Maximum Flows

We now present our algorithm SOCIAL, then we discuss its algorithmic components.

3.1 Overall Strategy

Given a graph $G = (V, E)$, a seed node u, and a motif μ, our strategy for local clustering is based on the following phases. First, we select a set $S \subseteq V$ containing u and close-by nodes. From now on, we refer to this set S as a *ball around* u. Second, we enumerate the collection M of occurrences of the motif μ which contain at least one node in S. Third, we build a hypergraph model H_μ in such a way that the motif-conductance of any cluster $C \subseteq S$ in G can be computed directly in H_μ. Fourth, we set $C_0 = S$ as our initial cluster and use it to build our MQI-based [26] flow model $\mathcal{N}$ from the hypergraph model H_μ. Fifth, we use $\mathcal{N}$ to either find a new cluster $C \subset C_0$ containing u with strictly smaller motif conductance than C_0 or prove that such cluster does not exist. While $C \subset C_0$ is found, we take it as our new initial cluster, rebuild $\mathcal{N}$, and repeat the previous phase. When eventually no such strict sub-set is found, the best obtained cluster is directly translated back to G as a local cluster around the seed node. Figure 2 provides a comprehensive illustration of the consecutive phases of SOCIAL. Note that there is no guarantee of finding the best overall cluster including u strictly contained in S. Instead, we find a succession of clusters with strictly decreasing cardinality and motif conductance until a local optimum is reached. To better explore the vicinity of u in G and overcome the fact we only find clusters inside S, we repeat the overall strategy α times with different balls S. Our overall algorithm including the mentioned repetitions is outlined in Algorithm 1.

3.2 Hypergraph Model

We follow the same procedure as Chhabra et al. [7] to construct the hypergraph model H_μ. To ensure a thorough understanding of our overall algorithm, we provide a summary of the phases involved, i.e., finding a ball around the seed node, enumerating motifs within it, and finally constructing the hypergraph model H_μ.

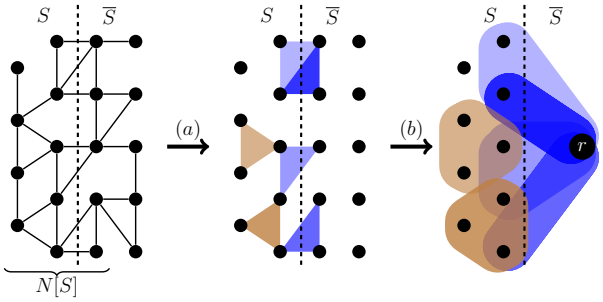


Figure 3 Example of motif-enumeration and model-construction phases for the triangle motif. In the left, the nodes of G are split into sets S and $\overline{S}$. In the center, motifs containing nodes in S are enumerated. In the right, H_μ is built by converting motifs in nets and contracting $\overline{S}$ into a node r.

Algorithm 1 Local Motif Clustering via Max Flows.

Input graph $G = (V, E)$; seed node $u \in V$; motif μ
Output cluster $C^* \subseteq V$

$C^* \leftarrow \emptyset$
for $i = 1, \ldots, \alpha$ **do**
  Select ball S around u
  $M \leftarrow$ Enumerate motifs in S
  Build hypergraph model H_μ based on S and M
  $C \leftarrow S$
  do
    $C_0 \leftarrow C$
    Build flow model $\mathcal{N}$ based on C_0 in H_μ
    Solve $\mathcal{N}$ to obtain cluster $C \subseteq C_0$ including u
  while $C \subset C_0$
  if $C^* = \emptyset \vee \phi_\mu(C) < \phi_\mu(C^*)$ **then**
    $C^* \leftarrow C$
Convert C^* into a local motif cluster in G

Ball around the Seed Node. Our approach to select a ball S is a fixed-depth breadth-first search (BFS) rooted on u. More specifically, we compute the first ℓ layers of the BFS tree rooted on u, then we include all its nodes in S. For each of the α repetitions of the overall algorithm, we use different amounts ℓ of layers for a better algorithm exploration. Two special cases are handled by `SOCIAL`, namely a ball S that is either too small or disconnected from $\overline{S}$. We avoid the first special case by ensuring that S contains 100 or more nodes in at least one repetition of our overall algorithm. More specifically, in case this condition is not automatically met, then we accomplish it in the last repetition by growing additional layers in our partial BFS tree while it contains fewer than 100 nodes. The number 100 is based on the findings of Leskovec et al. [29], which show that most well characterized communities from real-world graphs have a relatively small size, in the order of magnitude of 100 nodes. If the second exceptional case happens, it means that the whole BFS tree rooted on the seed node has at most ℓ layers. In this case, we simply stop the algorithm and return the entire ball S, which corresponds to an optimal community with motif conductance 0 provided that there is at least one motif in S and another one in $\overline{S}$. The number α of repetitions as well as the amount ℓ of layers used in each repetition are tuning parameters.

Motif Enumeration. Although enumerating a general motif on some graph is NP-hard [37], there are efficient heuristics to do it such as the one proposed by Kimmig et al. [23]. Nevertheless, simpler motifs such as small paths, cycles, and cliques can be trivially enumerated in

polynomial time. We focus our enumeration phase on the triangle motif. We implement the simple and exact algorithm proposed by Chiba and Nishizeki [8] to enumerate the collection M of occurrences of the motif μ which contain at least one node in S. Roughly speaking, this algorithm works by intersecting the neighborhoods of adjacent nodes. For each node v, the algorithm starts by marking its neighbors with degree smaller than or equal to its own degree. For each of these specific neighbors of v, it then scans its neighborhood and enumerates new triangles as soon as marked nodes are found. The running time of this algorithm is $O(ma)$, where a is the arboricity of the graph. We apply this enumeration algorithm only on the subgraph induced in G by $N[S]$, which is enough to find all triangles containing at least one node in S, as exemplified by transformation (a) in Figure 3. Assuming a constant-bounded arboricity, the overall cost of our motif-enumeration phase for triangles is $O\big(|N[S] \times N[S]) \cap E|\big)$.

Hypergraph Model. The hypergraph model H_μ is finally built in two conceptual operations. First, define a hypergraph containing V as nodes and a set $\mathcal{E}$ of nets such that, for each motif in M, $\mathcal{E}$ has a net with pins equal to the endpoints of this motif. Then, we contract together all nodes in $\overline{S}$ into a single node r and substitute parallel nets by a single net whose weight is equal to the summed weights of the removed parallel nets. More formally, we define the hypergraph version of our model as $H_\mu = (S \cup \{r\}, \mathcal{E})$ where the set $\mathcal{E}$ of nets contains one net e associated with each motif occurrence $G' = (V', E') \in M$ such that $e = V'$ if $V' \subseteq S$, and $e = V' \cap S \cup \{r\}$ otherwise. In the former case the net has weight 1, in the latter case the net has weight equal to the amount of motif occurrences in M represented by it. Since node weights in H_μ are irrelevant for `SOCIAL`, the involved theorems, and the motif conductance metric, we make all node weights unitary in H_μ. In practice, the model H_μ can be built by instantiating the nodes in $S \cup \{r\}$ and the nets in $\mathcal{E}$. Assuming that the number of nodes in μ is a constant, our model is built in time $O(|S| + |M|)$ and uses memory $O(|S| + |M|)$. The construction of H_μ is illustrated in transformation (c) of Figure 2 and demonstrated for a particular example in transformation (b) of Figure 3. Theorem 1 shows that the motif conductance in G of any cluster $C \subseteq S$ can be directly computed from H_μ assuming $d_\mu(S) \leq d_\mu(\overline{S})$. The assumption $d_\mu(S) \leq d_\mu(\overline{S})$ is fair in practice since the ball S computed via BFS tends to be considerably smaller than $\overline{S}$ for huge sparse networks. Enumerating the motifs in $\overline{S}$ is not reasonable for a local clustering algorithm, but we did verify that our assumption holds during all our experiments.

▶ **Theorem 1** (Theorem 3.2 from [7]). *The motif conductance $\phi_\mu(C)$ of a cluster $C \subseteq S$ in the original graph G can be calculated directly in the hypergraph model H_μ using the ratio of its cut-net $cut(C)$ to its weighted volume $d_w(C)$, assuming that the motif enumeration step is exact and $d_\mu(S) \leq d_\mu(\overline{S})$.*

3.3 Flow Model

In this section, we describe the process of constructing our `MQI`-based flow model $\mathcal{N}$ using the hypergraph model H_μ and an initial cluster $C_0 \subseteq S$ which contains the seed node u. There are three possible implementations of $\mathcal{N}$ based on the three already explained techniques to represent hypergraphs using graphs, namely *clique expansion*, *star expansion*, and *Lawler expansion* (see Figure 1). We show a bijective correspondence between certain s-t cuts in $\mathcal{N}$ and clusters $C \subseteq C_0$ in G that include the seed node u and have motif conductance less than that of C_0. We start by converting our hypergraph model H_μ in a directed graph using the chosen net expansion technique. Second, we find a corresponding cluster C_0'

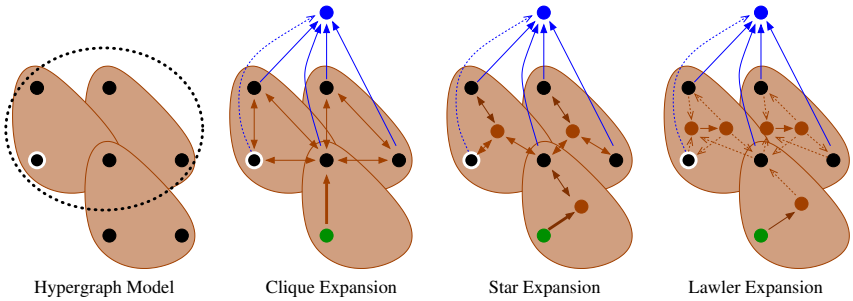


Figure 4 Flow model $\mathcal{N}$ given a hypergraph model H_μ and an initial cluster C_0. Nodes and nets of H_μ are respectively represented by black circles and brown areas around them. The seed node u is circled in white and the initial cluster C_0 is surrounded by a dotted ellipse. Auxiliary artificial nodes and edges used in each net-expansion are respectively represented by brown circles and arrows. Bidirectional arrows represent pairs of edges in both directions. The seed node s, the sink node t, and the in-edges of t are respectively represented by a green circle, a blue circle, and blue arrows. Solid and dashed arrows respectively represent edges with finite and infinite weight.

for C_0 in the created graph. For the *clique expansion*, $C'_0 = C_0$ since this transformation does not create artificial nodes. For the *star expansion*, C'_0 consists of C_0 and also the auxiliary artificial nodes connected to at least one node in C_0. For the *Lawler expansion*, C'_0 consists of C_0, the auxiliary artificial nodes w_1 having in-edges only from nodes in C_0, and the auxiliary artificial nodes w_2 having out-edges to at least one node in C_0. Third, we contract $\overline{C'_0}$ to a single *source* node s and then remove all its in-edges. Fourth, we multiply the weight of all the remaining edges by $d_{\mathcal{W}}(C_0)$, i.e., the weighted volume of C_0 in H_μ. Fifth, we introduce a sink node t and include in-edges to it from each of the nodes $v \in C_0 \setminus \{u\}$, such that the weight of (v,t) is set to $cut(C_0)d_{\mathcal{W}}(v)$, i.e., the cut-net of C_0 in H_μ multiplied by the weighted degree of v in H_μ. Finally, we include an edge (u,t) from the seed node to the sink and set its weight to infinity. Our flow network model $\mathcal{N}$ is concluded by setting edge capacities to match edge weights. Figure 4 shows the three possible configurations of our flow model $\mathcal{N}$ for a given hypergraph model H_μ and an initial cluster C_0.

We now analyze the theoretical guarantees provided by the defined flow model $\mathcal{N}$. Theorem 2 shows that there is a set $C \subset C_0$ in G including the seed node u with motif conductance smaller than that of C_0 if, and only if, the value of the maximum flow on $\mathcal{N}$ is less than $cut(C_0)d_{\mathcal{W}}(C_0)$, which is the weight of the trivial cut $(\{s\}, V(\mathcal{N}) \setminus \{s\})$. Due to space constraints, we put the proof of Theorem 2 in our public technical report [6]. In the proof, we show that such improved cluster C consists of the sink side of the minimum cut associated with the maximum flow. This cluster can be directly obtained with a reverse breadth-first search on $\mathcal{N}$ starting at the sink node. For an even stronger claim, see our public technical report [6]. Assumptions (a) and (b) in Theorem 2 are the same used in Theorem 1, which were previously shown to be reasonable in practice. Note that the claim is only valid for motifs with three nodes for clique and star expansion models, while it is valid in general for the Lawler expansion model.

▶ **Theorem 2.** *There is a set $C \subset C_0$ in G including the seed node u with motif conductance smaller than that of C_0 if, and only if, the maximum flow on $\mathcal{N}$ is less than $cut(C_0)d_{\mathcal{W}}(C_0)$ under the following assumptions:*

a) *the motif enumeration phase is exact;*

b) *$d_\mu(S) \leq d_\mu(\overline{S})$ in G;*

c) *in case $\mathcal{N}$ is based on clique expansion or star expansion, the motif μ has three nodes;*

Table 1 Graphs for experiments.

Graph	n	m	# Triangles
com-amazon	334 863	925 872	667 129
com-dblp	317 080	1 049 866	2 224 385
com-youtube	1 134 890	2 987 624	3 056 386
com-livejournal	3 997 962	34 681 189	177 820 130
com-orkut	3 072 441	117 185 083	627 584 181
com-friendster	65 608 366	1 806 067 135	4 173 724 142

SOCIAL utilizes a push-relabel approach to iteratively search for a maximum s-t flow in the model $\mathcal{N}$. If the found maximum flow is strictly smaller than $cut(C_0)\mathrm{d}_{\mathfrak{W}}(C_0)$, then we can directly find a minimum cut with the same weight as it and, consequently, a cluster $C \subset C_0$ containing the seed node u that has a strictly smaller motif conductance value $\phi_\mu(C)$ than that of C_0 in G. If such a cut is found, the algorithm repeats the process recursively setting the identified sub-cluster C as the new initial cluster, i.e., it constructs a new flow model based on H_μ and the initial cluster and uses the push-relabel algorithm to continue searching for sub-clusters with even lower motif conductance values. If, on the other hand, the maximum flow is not strictly smaller than $cut(C_0)\mathrm{d}_{\mathfrak{W}}(C_0)$, it means that the current cluster C_0 is optimal among all of its sub-clusters containing the seed node u, and the algorithm terminates for the given ball S.

4 Experimental Evaluation

Methodology. We implemented SOCIAL in C++. We compiled our program using gcc 11.2 with full optimization turned on (-O3 flag). All our experiments are based on the triangle motif, i.e., the undirected clique of size three. Since this motif has three nodes, Theorem 2 is valid for all net expansion techniques. Therefore, we focus our experiments on the clique expansion technique, which is more efficient than the other techniques because it does not utilize any auxiliary artificial nodes and uses the minimum amount of auxiliary artificial edges. We use the following parameters for SOCIAL: $\alpha = 3$, $\ell \in \{1, 2, 3\}$. We do not utilize more than 3 BFS layers because otherwise the ball around the seed node could become too large in densely connected areas of a graph. Nevertheless, as explained in Section 3.2, SOCIAL might occasionally use more than 3 BFS layers to ensure that the ball includes a minimum of 100 nodes, if the seed node under consideration is located in a sparse portion of the graph. We ensure the integrity of our results by using the same motif-conductance evaluator function for all tested algorithms. In our experiments, we have used a machine with a sixty-four-core AMD EPYC 7702P processor running at 2.0 GHz, 1 TB of main memory, 32 MB of L2-Cache, and 256 MB of L3-Cache. We measure running time, motif-conductance, and/or size of the computed cluster. For each graph, we pick 50 random seed nodes and use all of them as input for each algorithm. When averaging running time or cluster size over multiple instances, we use the geometric mean in order to give every instance the same influence on the *final score*. When averaging motif conductance over multiple instances, the final score is computed via arithmetic mean. This is a necessary averaging strategy since motif conductance can be zero, which makes the geometric mean infeasible to compute. We also use *performance profiles* which relate the running time (resp. motif conductance) of a group of algorithms to the fastest (resp. best) one on a per-instance basis. Their x-axis shows a factor τ while their y-axis shows the percentage of instances for which algorithm A has up to τ times the running time (resp. motif conductance) of the fastest (resp. best) algorithm.

Table 2 Average comparison against state-of-the-art. Times are given in seconds.

Graph	SOCIAL			LMCHGP			MAPPR		
	ϕ_μ	$\lvert C\rvert$	t(s)	ϕ_μ	$\lvert C\rvert$	t(s)	ϕ_μ	$\lvert C\rvert$	t(s)
com-amazon	**0.031**	76	<0.01	0.037	64	0.22	0.153	58	2.68
com-dblp	**0.090**	58	0.02	0.115	56	0.38	0.289	35	3.04
com-youtube	**0.125**	1832	4.52	0.172	1443	7.93	0.910	2	10.44
com-livejournal	**0.158**	494	3.33	0.244	387	8.17	0.507	61	173.80
com-orkut	0.273	1041	256.21	**0.150**	13168	496.94	0.407	511	923.26
com-friendster	0.388	2060	1194.50	**0.368**	10610	1339.99	0.741	121	16565.99
Overall	**0.178**	453	2.33	0.181	823	12.67	0.500	50	79.34

Instances. The graphs used in our experiments are the same ones used by Yin et al. [49] and Chhabra et al. [7] and the seed nodes used in our experiments are the same used in [7]. Specifically, we use real graphs from the SNAP Network Dataset Collection [28]. Prior to our experiments, we removed parallel edges, self-loops, and directions of edges and assigning unitary weight to all nodes and edges. Basic properties of the graphs under consideration can be found in Table 1.

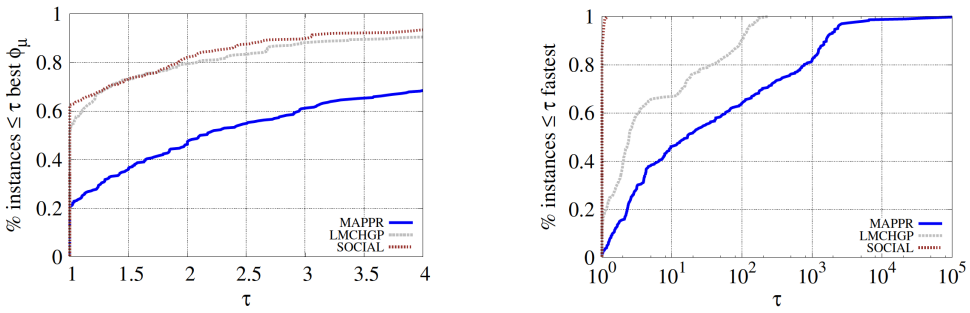
Competitors. We experimentally compare our SOCIAL against the state-of-the-art competitors, namely MAPPR [49] and the algorithm proposed by Chhabra et al. [7]. For conciseness, we refer to the latter one from now on as LMCHGP, an acronym for *local motif clustering via (hyper)graph partitioning*. We also ran preliminary experiments with HOSPLOC [51]. However, the algorithm is very slow even for small graphs and not scalable as their algorithm works using an adjacency matrix and hence needs $\Omega(n^2)$ space and time. Moverover, experiments done in their paper are on graphs that are multiple orders of magnitude smaller than the graphs used in our evaluation. Hence, we are not able to run the algorithm on the scale of the instances used in this work. We were not able to explicitly compare against LCD-Motif [50] since their code is not available (neither public, nor privately[1]) and the data presented in the respective paper does not warrant explicit comparisons (e.g. seed nodes are typically not presented in the papers). However, we try to make implicit comparisons in Section 4.1.

We compare our results against the globally best cluster computed for each seed node by MAPPR using its standard parameters ($\alpha = 0.98$, $\epsilon = 10^{-4}$) and by LMCHGP using the configuration with best overall results in [7] (graph model, label propagation, $\alpha = 3$, $\ell \in \{1, 2, 3\}$, and $\beta = 80$). Unless mentioned otherwise, experiments presented here involve all graphs from Table 1.

4.1 Results

The performance profile plots shown in Figures 5a and 5b, compare LMCHGP [7] and MAPPR [49] against SOCIAL. In Table 2, we show average results for each graph in our Test Set as well as average results overall. As shown in Figure 5a, SOCIAL obtains the best or equal motif conductance value for 62% of the instances, while LMCHGP and MAPPR respectively obtain the best or equal motif conductance for 49% and 19% of the instances. This result can be explained with two observations. First, SOCIAL explores the solution space considerably better than MAPPR, since we build our model multiple times, while MAPPR simply uses the APPR algorithm. Second, SOCIAL is based on a flow approach which directly optimizes for motif conductance, whereas LMCHGP is based on a (hyper)graph partitioning algorithm which

[1] Personal communication with the authors



(a) Motif conductance. **(b)** Running time.

Figure 5 Performance profiles for motif conductance and running time considering all instances, i.e., 50 random seed nodes for each graph in Table 1. In the running time plot, the `SOCIAL` curve is roughly coincident with the y-axis.

is repeated multiple times to compensate for its design to minimize the number of cut motifs rather than motif conductance. In Table 2, `SOCIAL` outperforms `LMCHGP` for 4 of the 6 graphs and overall, and outperforms `MAPPR` for all graphs and overall. Overall, `SOCIAL` computes clusters with motif conductance 0.178 while `LMCHGP` and `MAPPR` compute clusters with motif conductance 0.181 and 0.500, respectively.

As exhibited in Figure 5b, `SOCIAL` is the fastest one for 87% of the instances, while `LMCHGP` and `MAPPR` are the fastest ones for 12% and 1% of the instances, respectively. Furthermore, the running time of `SOCIAL` is within a factor 1.18 of the running times of the fastest competitors for all instances. `SOCIAL` is respectively up to 237 and 144 063 times faster than `LMCHGP` and `MAPPR` while being a factor 5.4 and 34.1 faster than them on average. The reason for `MAPPR` being considerably slower than the other algorithms is that it must enumerate motifs across the entire graph, while `SOCIAL` and `LMCHGP` only require enumeration of motifs in a ball around the seed node. The reduced but still substantial difference in running time between `SOCIAL` and `LMCHGP` is a result of `LMCHGP`'s repeated partitioning of each ball around the seed node, while `SOCIAL` employs a flow model to greedily improve the motif conductance metric until a local optimum cluster is obtained. In Table 2, `SOCIAL` outperforms `LMCHGP` and `MAPPR` on average in terms of running time for every single graph and overall.

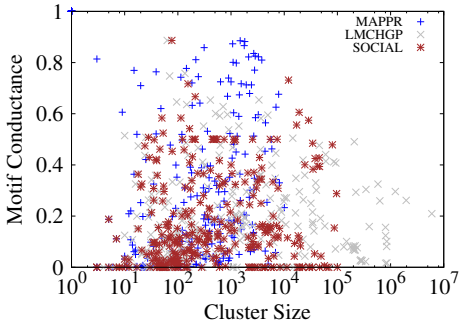


Figure 6 Motif conductance vs cluster size.

For a more intuitive analysis of the quality of our results, Figure 6 plots motif conductance versus cluster size for all communities computed by the three algorithms. Observe that the communities found by SOCIAL are densely localized in the lower left area of the chart, which is the region with smaller motif conductance and smaller cluster size. On the other hand, communities computed by MAPPR are often in the upper area of the chart and communities computed by LMCHGP are often in the right area of the chart.

Additional Comparisons. As we mention above, we were not able to run comparisons against LCD-Motif [50] explicitly since their code is not available (neither publicly, nor privately) and the data presented in the respective papers does not warrant explicit comparisions (e.g., seed nodes are typically not presented in papers, and in this case instances are directed rather than undirected). Here, we make an attempt at implicit comparisons. Zhang et al. [50] (Table 4 therein) compare motif conductance against MAPPR on three directed instances (cit-HepPh, Slashdot, StanfordWeb) and report an geometric mean improvement of 54% in motif conductance for the triangle motif. As SOCIAL works for undirected instances, we have build undirected version of those graphs and run SOCIAL as well as MAPPR for the triangle motif. The geometric mean improvement we obtain over MAPPR is 223% which is significantly larger than the improvement of Zhang et al. [50] over MAPPR. Also note that in our experiments from Table 2, the geometric mean improvement (using the average motif conductance values) of SOCIAL over MAPPR in motif conductance is 219%.

5 Conclusion

In this work, we propose SOCIAL, a fast flow-based algorithm to solve the local motif clustering problem in graphs. Given a seed node, our SOCIAL selects a ball of nodes around it, which is taken as an initial cluster and used to build an exact hypergraph model where nets represent motifs. Using this model and the initial cluster, we create a flow model in which the value of the maximum s-t flow is directly related to the presence of sub-sets of the initial cluster that contain the seed node and have lower motif conductance than the initial cluster as a whole. Utilizing a push-relabel algorithm, SOCIAL either identifies a sub-cluster containing the seed node with improved motif conductance and repeats the process recursively by considering it as the initial cluster, or demonstrates that the current initial cluster is the best among all its sub-clusters that include the seed node.

In our experiments with the triangle motif, we found that SOCIAL produces communities with an average motif conductance better than the state-of-the-art, while running up to orders of magnitude faster on average. For future work, we intend to conduct experiments with larger motifs and use the Lawler-expansion version of our flow model, since it is the only one whose quality guarantee holds true for larger motifs. Laslty, we intend to add parallelization to improve the speed on large instances further.

References

1 Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486, 2006. doi:10.1109/FOCS.2006.44.

2 Reid Andersen and Kevin J. Lang. An algorithm for improving graph partitions. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 651–660. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347154.

3 Austin R. Benson, David F. Gleich, and Jure Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 118–126. SIAM, 2015. doi:10.1137/1.9781611974010.14.

4 Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. doi:10.1126/science.aad9029.

5 Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Trans. Knowl. Data Eng.*, 20(2):172–188, 2008. doi:10.1109/TKDE.2007.190689.

6 Adil Chhabra, Marcelo Fonseca Faraj, and Christian Schulz. Faster local motif clustering via maximum flows. *CoRR*, abs/2301.07145, 2023. doi:10.48550/arXiv.2301.07145.

7 Adil Chhabra, Marcelo Fonseca Faraj, and Christian Schulz. Local motif clustering via (hyper)graph partitioning. In *Symposium on Algorithm Engineering and Experiments (ALENEX 23), January 22-23, 2023*. SIAM, 2023.

8 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comp.*, 14(1):210–223, 1985. doi:10.1137/0214017.

9 Fan Chung and Olivia Simpson. Solving linear systems with boundary conditions using heat kernel pagerank. In *Intl. Workshop on Algorithms and Models for the Web-Graph*, pages 203–219. Springer, 2013. doi:10.1007/978-3-319-03536-9_16.

10 Jonathan Cohen. Trusses: Cohesive subgraphs for social network analysis. *National security agency Tech. report*, 16(3.1), 2008. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.505.7006&rep=rep1&type=pdf.

11 Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. Local search of communities in large graphs. In *ACM SIGMOD Intl. Conf. on Management of data*, pages 991–1002, 2014. doi:10.1145/2588555.2612179.

12 Alessandro Epasto, Jon Feldman, Silvio Lattanzi, Stefano Leonardi, and Vahab Mirrokni. Reduce and aggregate: similarity ranking in multi-categorical bipartite graphs. In *WWW*, pages 349–360, 2014. doi:10.1145/2566486.2568025.

13 Lester Randolph Ford and Delbert Ray Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

14 Kimon Fountoulakis, Pan Li, and Shenghao Yang. Local hyper-flow diffusion. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27683–27694, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/e924517087669cf201ea91bd737a4ff4-Abstract.html.

15 Kimon Fountoulakis, Meng Liu, David F. Gleich, and Michael W. Mahoney. Flow-based algorithms for improving clusters: A unifying framework, software, and performance. *SIAM Rev.*, 65(1):59–143, 2023. doi:10.1137/20m1333055.

16 Andrew V. Goldberg and Robert Endre Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988. doi:10.1145/48014.61051.

17 Lars Gottesbüren, Tobias Heuer, Peter Sanders, and Sebastian Schlag. Scalable shared-memory hypergraph partitioning. In *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX*, pages 16–30. SIAM, 2021. doi:10.1137/1.9781611976472.2.

18 Lars Gottesbüren, Tobias Heuer, Peter Sanders, Christian Schulz, and Daniel Seemaier. Deep multilevel graph partitioning. In *29th Annual European Symposium on Algorithms, ESA*, volume 204 of *LIPIcs*, pages 48:1–48:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.48.

19 Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. Querying k-truss community in large and dynamic graphs. In *ACM SIGMOD*, pages 1311–1322, 2014. doi:10.1145/2588555.2610495.

20 Rania Ibrahim and David F. Gleich. Local hypergraph clustering using capacity releasing diffusion. *CoRR*, abs/2003.04213, 2020. arXiv:2003.04213.

21 Lucas G. S. Jeub, Prakash Balachandran, Mason A. Porter, Peter J. Mucha, and Michael W. Mahoney. Think locally, act locally: Detection of small, medium-sized, and large communities in large networks. *Physical Review E*, 91(1):012821, 2015. doi:10.1103/PhysRevE.91.012821.

22 Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *JACM*, 51(3):497–515, 2004. doi:10.1145/990308.990313.

23 Raphael Kimmig, Henning Meyerhenke, and Darren Strash. Shared memory parallel subgraph enumeration. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops*, pages 519–529. IEEE Computer Society, 2017. doi:10.1109/IPDPSW.2017.133.

24 Kyle Kloster and David F. Gleich. Heat kernel based community detection. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*, pages 1386–1395. ACM, 2014. doi:10.1145/2623330.2623706.

25 Christine Klymko, David F. Gleich, and Tamara G. Kolda. Using triangles to improve community detection in directed networks. *CoRR*, abs/1404.5874, 2014. arXiv:1404.5874.

26 Kevin J. Lang and Satish Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *Integer Programming and Combinatorial Optimization, 10th International IPCO*, volume 3064 of *LNCS*, pages 325–337. Springer, 2004. doi:10.1007/978-3-540-25960-2_25.

27 Eugene L. Lawler. Cutsets and partitions of hypergraphs. *Networks*, 3(3):275–285, 1973. doi:10.1002/net.3230030306.

28 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

29 Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Math.*, 6(1):29–123, 2009. doi:10.1080/15427951.2009.10129177.

30 Yixuan Li, Kun He, David Bindel, and John E. Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In *Proceedings of the 24th International Conference on World Wide Web, WWW*, pages 658–668. ACM, 2015. doi:10.1145/2736277.2741676.

31 Meng Liu, Nate Veldt, Haoyu Song, Pan Li, and David F. Gleich. Strongly local hypergraph diffusions for clustering and semi-supervised learning. In *WWW '21: The Web Conference*, pages 2092–2103. ACM / IW3C2, 2021. doi:10.1145/3442381.3449887.

32 Michael W. Mahoney, Lorenzo Orecchia, and Nisheeth K. Vishnoi. A local spectral method for graphs: with applications to improving graph partitions and exploring data graphs locally. *J. Mach. Learn. Res.*, 13:2339–2365, 2012. doi:10.5555/2503308.2503318.

33 Tao Meng, Lijun Cai, Tingqin He, Lei Chen, and Ziyun Deng. Local higher-order community detection based on fuzzy membership functions. *IEEE Access*, 7:128510–128525, 2019. doi:10.1109/ACCESS.2019.2939535.

34 Mrudula Murali, Katerina Potika, and Chris Pollett. Online local communities with motifs. In *2020 Second Intl. Conf. on Transdisciplinary AI (TransAI)*, pages 59–66. IEEE Computer Society, September 2020. doi:10.1109/TransAI49837.2020.00014.

35 Lorenzo Orecchia and Zeyuan Allen Zhu. Flow-based algorithms for local graph clustering. In *SODA*, pages 1267–1286. SIAM, 2014. doi:10.1137/1.9781611973402.94.

36 Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007. doi:10.1093/bioinformatics/btl301.

37 Ronald C. Read and Derek G. Corneil. The graph isomorphism disease. *J. Graph Theory*, 1(4):339–363, 1977. doi:10.1002/jgt.3190010410.

38 Karl Rohe and Tai Qin. The blessing of transitivity in sparse and stochastic networks. *arXiv preprint*, 2013. arXiv:1307.2302.

39 Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *Algorithms – ESA 2011 – 19th Annual European Symposium*, volume 6942 of *LNCS*, pages 469–480. Springer, 2011. doi:10.1007/978-3-642-23719-5_40.

40 Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, and Christian Schulz. k-way hypergraph partitioning via n-level recursive bisection. In *Proceedings of the Workshop on Algorithm Engineering and Experiments, ALENEX*, pages 53–67. SIAM, 2016. doi:10.1137/1.9781611974317.5.

41 Ronghua Shang, Weitong Zhang, Jingwen Zhang, Jie Feng, and Licheng Jiao. Local community detection based on higher-order structure and edge information. *Physica A: Statistical Mechanics and its Applications*, 587:126513, 2022. doi:10.1016/j.physa.2021.126513.

42 Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 939–948. ACM, 2010. doi:10.1145/1835804.1835923.

43 Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comput.*, 42(1):1–26, 2013. doi:10.1137/080744888.

44 Charalampos E. Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web, WWW*, pages 1451–1460. ACM, 2017. doi:10.1145/3038912.3052653.

45 Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Minimizing localized ratio cut objectives in hypergraphs. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1708–1718. ACM, 2020. doi:10.1145/3394486.3403222.

46 Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Hypergraph cuts with general splitting functions. *SIAM Rev.*, 64(3):650–685, 2022. doi:10.1137/20m1321048.

47 Konstantin Voevodski, Shang-Hua Teng, and Yu Xia. Spectral affinity in protein networks. *BMC systems biology*, 3(1):1–13, 2009. doi:10.1186/1752-0509-3-112.

48 Dorothea Wagner and Frank Wagner. Between min cut and graph bisection. In *Mathematical Foundations of Computer Science 1993, International Symposium, MFCS*, volume 711 of *LNCS*, pages 744–750. Springer, 1993. doi:10.1007/3-540-57182-5_65.

49 Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564. ACM, 2017. doi:10.1145/3097983.3098069.

50 Yunlei Zhang, Bin Wu, Yu Liu, and Jinna Lv. Local community detection based on network motifs. *Tsinghua Science and Technology*, 24(6):716–727, 2019. doi:10.26599/TST.2018.9010106.

51 Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. High-order structure exploration on massive graphs: A local graph clustering perspective. *ACM Trans. Knowl. Discov. Data*, 15(2):18:1–18:26, 2021. doi:10.1145/3425637.