Efficient Edge Weight Estimation for Motif-based Graph Partitioning: An Adaptive Sampling Approach

Shixun Huang
RMIT University

Yuchen Li
Singapore Management University

Zhifeng Bao
RMIT University

ABSTRACT

In this paper, we study the problem of efficient and scalable motif-based graph partitioning (MGP), where existing methods usually follow a two-step workflow: first, the exact weight of each edge is computed by counting the number of induced subgraphs that are isomorphic to the query motif and also contain this edge; second, the computed weighted graph is fed into an existing Graph Partitioning (GP) kernel to produce the partitioning result. After showing that the first step is the main bottleneck, we first propose a sampling-based MGP (SMGP) approach that employs an unbiased sampling mechanism to efficiently estimate the edge weights while trying to preserve the partitioning quality. To further improve the effectiveness, we propose a novel adaptive sampling framework called SMGP+. SMGP+ iteratively partitions the input graph based on up-to-date estimated edge weights, and adaptively adjusts the sampling distribution so that edges that are more likely to affect the partitioning outcome will be prioritized for weight estimation. To our best knowledge, this is the first attempt to solve the MGP problem without employing exact edge weight computations. Some highlights of our framework are as follows. First, it gives hope for existing MGP methods to perform on complicated motifs in a scalable yet effective manner. Second, SMGP+ is general to work with different underlying GP kernels adopting popular partition quality metrics, such as modularity, edge-cut weight and conductance. Third, we prove that both SMGP and SMGP+ are unbiased estimators and show a theoretical guarantee on the quality of result. Last, extensive experiments have validated that our framework delivers competitive partitioning quality compared to existing workflows based on exact edge weights, while achieving up to four orders of magnitude speedup.

PVLDB Reference Format:
Shixin Huang, Yuchen Li, Zhifeng Bao. Efficient Edge Weight Estimation for Motif-based Graph Partitioning: An Adaptive Sampling Approach. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

Graph Partitioning (GP) is a fundamental yet core operator in processing large graphs and has been used in a wide range of domains including community detection [37], distributed computing [23] and image processing [20]. While traditional GP approaches focus on reducing the number of plain edges between the partitions, recent studies have unveiled that the network motifs, defined by a particular pattern of interactions between vertices, may provide a deep insight into discovering high-order communities from complex networks [14, 13, 25, 39]. For example, triangular motifs are crucial to finding ground-truth communities in social networks [46] and identifying structural hubs in the brain [39], and multi-hop loops may indicate money laundering process in financial transactions [14].

Henceforth motif-based GP (MGP), which aims to preserve network motifs during the partitioning process, has attracted many attentions [46, 6, 5, 33, 38, 43]. To partition a graph G with a graph pattern Q (aka. the *query motif*), existing MGP approaches employ a two-step workflow. In the first step, G is transformed to an edge-weighted graph where the weight of each edge e is the number of induced subgraphs in G that are isomorphic to Q and contain e. In the second step, existing GP kernels are invoked to partition the weighted graph obtained from the first step. Those GP kernels can be further classified according to the objective function adopted. The choice of the objective function depends on the downstream applications, where three widely adopted ones are: *edge-cut weight* [29], *conductance* [3], and *modularity* [16].

Counting the number of induced subgraphs in G that are isomorphic to Q computationally intensive since determining whether G contains a subgraph that is isomorphic to Q is NP-complete [17]. Thus, the first step of the workflow above becomes the main efficiency bottleneck for large graphs. As evidenced by our experiments, it takes more than one month to compute the exact edge weights over a small-scale graph with a motif of 6 vertices in a single thread (i.e., on Gowalla with motif Q_5).

To mitigate the inefficiency of exact edge weight computations, we devise a sampling mechanism called SMGP which is an unbiased estimator of exact edge weights and tries to preserve the quality of partitioning result produced upon the estimated weights. SMGP is inspired by a recent sampling approach [36] for motif counting that focuses on estimating the total count of a motif appeared in G. In contrast, SMGP needs to cautiously estimate the motif counts for every edge in G, because inaccurate edge weight estima-

tions may lead to drastically different partitioning outcomes, making this problem more challenging. Hence we devise an *edge-centric sampling method*, which first samples a target edge e as an induced subgraph of size 2, and then expands the subgraph by iteratively sampling neighborhood vertices for edge weight estimations. Upon sampling a subgraph I that is isomorphic to the query Q, we update the weights of all edges in I and the estimated weighted graph is later fed to a GP kernel for partitioning the graph. We show that the edge-centric sampling method is an *unbiased estimator* for the exact edge weight, and thereafter establish the approximation guarantee of SMGP on the partitioning quality (in terms of the *edge-cut weight* metric).

SMGP treats each edge equally and may generate an unnecessarily large number of samples, in order to ensure an accurate weight estimation for every edge. However, edges have varying degrees of importance in affecting the outcomes of GP kernels. For example, edges in densely connected subgraphs are likely to have large exact edge weights but they can be assigned with small estimated weights due to inaccurate estimations. Thus it is very likely that the underlying GP kernel would place the endpoints of these edges into different partitions. In contrast, the accurate estimation for edges with small exact weights is relatively less crucial to producing high-quality partition results as the decision on placing these edges may have very limited influence on the partitioning quality.

In order to further improve the partitioning quality with limited budget (e.g., the sample size), we propose a novel *adaptive sampling framework* called SMGP+. In each iteration of SMGP+, it performs sampling for edge weight estimation followed by calling a GP kernel to partition the graph. The partitioning outcome from the previous iteration is used to adjust the sampling distribution for the next iteration, where edges that are likely to affect the partitioning outcome are prioritized for weight estimation. We prove that SMGP+ is also an *unbiased estimator* of exact edge weights based on *martingales*, a classical statistical tool. Extensive experiments on real-world networks have shown that SMGP+ achieves significantly better partitioning quality than SMGP given the same number of samples.

To our best knowledge, this is the first attempt to solve the MGP problem without employing exact edge weight computations, and our sampling based methods, for the first time, give hope for existing motif-based graph partitioning (MGP) methods to perform on complex-structured motifs in a scalable yet effective manner. Our contributions are summarized as follows:

1. We propose a sampling-based approach called SMGP for the MGP problem, and prove that SMGP is an unbiased estimator of exact edge weights.
2. We propose an adaptive sampling framework SMGP+ that further improves the effectiveness of SMGP, and show that SMGP+ is also an unbiased estimator of exact edge weights. SMGP+ can work with multiple underlying GP methods adopting all popular evaluation metrics (i.e., edge-cut weight, modularity and conductance).
3. We prove the theoretical guarantees on the partitioning quality based on the weighted graph produced by SMGP and SMGP+ respectively, when the objective of the underlying GP method is edge-cut weight minimization.

Notation	Description
$V(G)$	The set of vertices in graph G.
$E(G)$	The set of edges in graph G.
N_v	The set of neighbors of v (in G).
$G' \simeq G$	Graph G' is isomorphic to graph G.
t	The number of iterations.
θ	The sample size in each iteration.
k	The number of partitions.
ϕ_k	A partitioning plan with k partitions.
$\phi_k(u)$	The ID of the partition to which vertex u belongs.
φ_i	The partition with an ID i.
w_e^i	The estimated weight of edge e in the i-th iteration.
w_e^{avg}	The average estimated weight of edge e across partitions.
$\pi(I)$	The probability of sampling the induced subgraph I.
$\widetilde{\pi}(e)$	The probability of sampling edge e as the starting edge.
$\mathcal{C}(u, \varphi_i)$	The set of edges connecting u to vertices in partition φ_i.

Table 1: Frequently used notations.

4. We conduct extensive experiments and verify: (1) SMGP is able to produce competitive results compared with the one produced based on exact edge weights in some cases, while achieving up to four orders of magnitude speedup; (2) SMGP+ significantly outperforms SMGP and, in many cases, produces nearly-identical partitioning qualities as compared to the one based on exact edge weights without notably compromising the efficiency. We refer readers to our technical report [1] for more experiment results.

Organization of this paper. The problem formulation is presented in Section 2. Next, we propose SMGP in Section 3, followed by SMGP+ in Section 4. The experimental results are discussed in Section 5. Related work is surveyed in Section 6. Last, we conclude the paper in Section 7.

2. PROBLEM FORMULATION

Given an undirected, connected and unweighted graph G, the set of vertices and edges of G are represented as $V(G)$ and $E(G)$ respectively. For a vertex $v \in V(G)$, N_v denotes the set of v's neighbors, and $|N_v|$ denotes the degree of v. A graph G' is called a *subgraph* of G if $V(G') \subset V(G)$ and $E(G') \subset E(G)$. A subgraph G' is an *induced subgraph* of G if $\forall\, u \in V(G), \forall\, v \in V(G)$ and $(u, v) \in E(G)$, we have $(u, v) \in E(G')$. A query motif Q is defined to be an undirected, unweighted and connected graph. Table 1 summarizes some frequently used notations in problem definitions and solutions.

Definition 1 (Isomorphism)**.** A graph G' is *isomorphic* to another graph G, denoted by $G' \simeq G$, if there is a bijection $f : V(G') \to V(G)$ such that $\forall u \in V(G)$, $\forall v \in V(G)$, $(u, v) \in E(G')$ if and only if $(f(u), f(v)) \in E(G)$.

Definition 2 (Instance of a Motif)**.** Given a graph G and a query motif graph Q, we call each induced subgraph of G that is isomorphic to Q as an *instance* of Q in G.

Definition 3 (Motif-based Weighted Graph)**.** Given a query motif Q and a graph G, we compute a *motif-based weighted graph* of G such that the weight w_e of each edge e represents the number of instances of Q in G that contain e.

Definition 4 (Graph Partition)**.** A *partition plan* ϕ_k of a graph G is a division of $V(G)$ into k disjoint vertex sets φ_i $(1 \le i \le k)$ such that $\cup_{i=1}^{k} \varphi_i = V(G)$. The partition

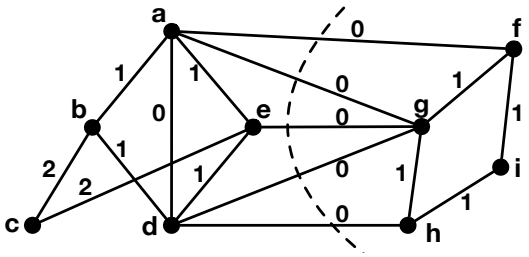


Figure 1: A weighted graph based on the motif ⊏.

ID assigned to a vertex v is denoted as $\phi_k(v)$. The size of a partition φ_i is denoted by $|\varphi_i|$ (i.e., the number of vertices assigned to φ_i).

Definition 5 (Edge-Cut Weight)**.** Given a partition plan ϕ_k of a graph G, the *edge-cut weight* of ϕ_k is the sum of the weights of the edges crossing the partitins, i.e.,

$$Cut(\phi_k) = \sum_{(u,v)\in E(G), \phi_k(u)\neq\phi_k(v)} w_{(u,v)}$$

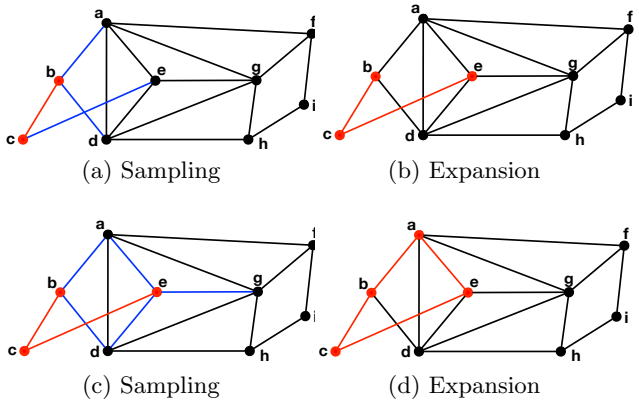
Definition 6 (Motif-based Graph Partition)**.** Given a query motif Q and a graph G, the goal of the motif-based graph partition (MGP) is to find a partition plan ϕ_k that minimizes the motif-based edge-cut weight, i.e., $\min_{\phi_k} Cut(\phi_k)$.

It is worth noting that there are three popular metrics for MGP's quality evaluation, namely, edge-cut weight [29], modularity [16] and conductance [3]. Modularity measures the density of edges inside partitions compared to crossing edges. Conductance measures the ratio of edge-cut weight over the sum of weights of edges in the smallest partition. It is not hard to see that edge-cut weight is a central factor in these objectives above. Thus, we design our approaches with the goal of minimizing edge-cut weight such that our approaches can easily work with multiple GP kernels with different objective functions.

Example 1. Figure 1 shows an example of how the exact edge weights are computed and an ideal partitioning result based on the query motif ⊏. The weight of an edge e indicates how many instances of ⊏ contain e. Here, the weight of (b, c) is 2 since it is contained in two instances of the query motif formed by vertices $\{a, b, c, e\}$ and $\{b, c, d, e\}$ respectively. The weight of some edges like (a, f) and (a, g) is 0 as they are not contained in any induced subgraphs isomorphic to ⊏. If we want to partition all vertices into two sets while preserving motif based communities, the ideal result is that the sum of weights of cross-partition edges is 0, as shown in the figure.

3. SAMPLING-BASED EDGE WEIGHT ESTIMATION

In this section, we introduce an edge-centric sampling method (in the first step of MGP), namely SMGP, to estimate the edge weights, which will be fed to a GP kernel (in the second step of MGP). In what follows, we first present the edge-centric sampling method, then prove its unbiased estimation property, and last discuss the theoretical guarantee of SMGP.



(a) Sampling (b) Expansion

(c) Sampling (d) Expansion

Figure 2: Subgraph expansion (best viewed in color).

3.1 Edge-centric Sampling

To estimate the edge weights in G, we need to compute how many instances of a given query motif Q contain any edge. A straightforward strategy is to uniformly sample many induced subgraphs of size $|V(Q)|$, then check whether they are instances of Q, and update the edge weights if so. However, this strategy is not efficient because the sampled subgraphs may be unconnected and an extremely large number of samples are needed for accurate estimations. To make sure that all sampled subgraphs are connected, we propose an edge-centric sampling strategy, which works by uniformly picking a starting edge (v_1, v_2) and then sampling the connected subgraphs expanded from the starting edges. In the rest of this section, we will introduce how we conduct edge-centric sampling via a process called *subgraph expansion*, to compute the subgraph sampling probabilities and finally estimate the edge weights.

Subgraph Expansion. We perform the subgraph expansion process by adopting a randomized procedure [35, 8, 36, 22, 45]. At the initial stage, we uniformly select a starting edge as an induced subgraph of size 2. Then we incrementally expand the size of the current induced subgraph S by adding a randomly selected vertex v as well as the edges that connect v and the vertices in S. We stop the expansion process once the size of S reaches $|V(Q)|$.

Example 2. Figure 2 shows an example of this expansion procedure. Here, the graph formed with red vertices and edges is the currently sampled subgraph and blue edges are involved in choosing a vertex in the neighborhood for expansion. Suppose we have a query motif ⊏. We first randomly sample a starting edge as the starting point for expansion. To decide which vertex in its neighborhood will be added into the current induced subgraph, we uniformly choose an edge (u, v) from all the blue edges that connect a vertex in the currently sampled subgraph to a vertex outside of this subgraph. If $u(v)$ is currently not in the sampled subgraph, we add $u(v)$ together with all edges connecting $u(v)$ and vertices in the currently sampled subgraph for expansion, which leads to a new induced subgraph of bigger size. We terminate the expansion once the size of the sampled subgraph is equal to the size of ⊏. In this example, the sampled subgraph is isomorphic to the query motif and it will contribute to the edge weight estimation which will be

introduced later.

Subgraph Sampling Probability. Given a query motif Q and a graph G, for each sampled subgraph of size $|V(Q)|$ in G, we can get a corresponding order of vertices $[v_1, v_2, ..., v_{|V(Q)|}]$ of the expansion procedure. Suppose we have a sampled subgraph S_i of size $i < |V(Q)|$ that is formed by the ordered vertices $[v_1, v_2, ..., v_i]$, then the probability of sampling the next vertex v_{i+1} is:

$$\mathbb{P}[v_{i+1}|S_i] = \frac{|E(S_{i+1})| - |E(S_i)|}{\sum_{j=1}^{i} |N_{v_j}| - 2|E(S_i)|} \tag{1}$$

Since v_{i+1} is uniformly sampled from the neighborhood of subgraph S_i, it then leads to the probability of sampling an induced subgraph $S_{|V(Q)|}$ as:

$$\begin{aligned}\pi(S_{|V(Q)|}) &= \frac{1}{|E(G)|} \prod_{i=2}^{|V(Q)|-1} \mathbb{P}[v_{i+1}|S_i] \\ &= \frac{1}{|E(G)|} \prod_{i=2}^{|V(Q)|-1} \frac{|E(S_{i+1})| - |E(S_i)|}{\sum_{j=1}^{i} |N_{v_j}| - 2|E(S_i)|}\end{aligned} \tag{2}$$

where $\frac{1}{|E(G)|}$ is the sampling probability of the starting edge (v_1, v_2).

Example 3. Suppose we have sampled an induced subgraph I that is an instance of motif ⊡ and is formed by the sequence of ordered vertices $[v_1, v_2, v_3, v_4]$, then the probability of sampling the instance I is:

$$\pi(I) = \frac{1}{|E(G)|} \frac{1}{|N_{v_1}| + |N_{v_2}| - 2} \frac{2}{|N_{v_1}| + |N_{v_2}| + |N_{v_3}| - 4}$$

Edge Weight Estimation. To this end, suppose we have sampled θ connected and induced subgraphs $I_1, I_2, ..., I_\theta$ through the above subgraph expansion process, then for any edge $e \in E(G)$, its weight can be estimated with the Horvitz-Thompson inverse probability weighting [26] as follows:

$$w_e = \frac{1}{\theta} \sum_{j=1}^{\theta} \frac{\mathbb{1}(e \in E(I_i)) \cdot \mathbb{1}(I_i \simeq Q)}{|\mathcal{O}(I_i)| \cdot \pi(I_i)} \tag{3}$$

Here, $\mathbb{1}$ refers to the indicator function, $|\mathcal{O}(I_j)|$ refers to the number of possible orderings of vertices that can lead to I_j, and $\pi(\cdot)$ refers to the subgraph sampling distribution. It is easy to see this is an unbiased estimation of edge weights so long as π is supported over all possible (connected) induced subgraphs of size $|V_Q|$. Note that $|\mathcal{O}(I_j)|$ is only dependent on the query motif Q and can be efficiently precomputed by dynamic programming.

3.2 Theoretical Analysis of SMGP

Intuitively, when the number of sampled subgraphs (aka. the sample size θ) is sufficiently large, we can get an accurate approximation such that each estimated edge weight is close to its exact edge weight. In such a case, we can feed the graph with the estimated weights to a GP kernel that produces a competitive partitioning result, compared with the one using exact edge weights. Next, we will show how the estimated weights and the partitioning quality converge to their expected values as the sample size θ grows, based on the Hoeffding's inequality [24] below.

Lemma 1. ([24]). Let $X_1, ..., X_\theta$ be θ independent and bounded random variables ($a_i \leq X_i \leq b_i$) and $\widetilde{X} = \frac{1}{\theta} \sum_{i=1}^{\theta} X_i$. For any $\epsilon \geq 0$, we have

$$\mathbb{P}[|\widetilde{X} - \mathbb{E}[\widetilde{X}] \geq \epsilon|] \leq \exp(-\frac{2\theta^2\epsilon^2}{\sum_{i=1}^{\theta}(b_i - a_i)^2})$$

Since each induced subgraph of size $|V(Q)|$ is sampled independently, we can apply Lemma 1 to give a concentration bound on estimating w_e (Equation 3) if we can bound the value of $X_i = \frac{\mathbb{1}(e \in E(I_i)) \cdot \mathbb{1}(I_i \simeq Q)}{|\mathcal{O}(I_i)| \cdot \pi(I_i)}$. The lower-bound of X_i is 0 if the sampled induced subgraph I_i is not isomorphic to the query motif. In order to decide the upper-bound, we need to calculate the lowest possible probability of sampling an isomorphic subgraph, and it is decided by $\pi(\cdot)$ (as defined in Equation 2) that satisfies the following inequality:

$$\begin{aligned}\pi(S_{|V(Q)|}) &= \frac{1}{|E(G)|} \prod_{i=2}^{|V(Q)|-1} \frac{|E(S_{i+1})| - |E(S_i)|}{\sum_{j=1}^{i} |N_{v_j}| - 2|E(S_i)|} \\ &> \frac{1}{|E(G)|} \cdot \frac{1}{\prod_{i=2}^{|V(Q)|-1} \sum_{j=1}^{i} |N_{v_j}|}\end{aligned}$$

Clearly, $\pi(S_{|V(Q)|})$ is lower bounded, which derives that $\frac{1}{\pi(\cdot)}$ is upper bounded. Thus, we use η to denote an upper bound on $\frac{1}{\pi(\cdot)}$. Since each weight estimation is independent and bounded, we can reach the following theorem.

Theorem 1. Given the weights $X_1, ..., X_\theta$ estimated for an edge e where $X_i = \frac{\mathbb{1}(e \in E(I_i)) \cdot \mathbb{1}(I_i \simeq Q)}{|\mathcal{O}(I_i)| \cdot \pi(I_i)}$, the sample size θ, and the upper-bound η of the estimated weights, then the deviation of the average of the estimated weights of e from $\mathbb{E}[w_e]$ is bounded with a probability increased w.r.t. θ, such that for any $\epsilon > 0$,

$$\mathbb{P}[|w_e - \mathbb{E}[w_e] \geq \epsilon|] \leq \exp(-\frac{2\theta\epsilon^2}{\eta^2})$$

where $w_e = \frac{1}{\theta} \sum_{i=1}^{\theta} X_i$.

Next, we study the impact of the sample size θ on the quality of partitions. It is worth noting that the sum of weights of all edges is fixed as a constant. Thus, the problem of minimizing the edge-cut weight can be transformed to the problem of maximizing the sum of weights of non-crossing edges. Given a partition plan ϕ_k, let $z'(\phi_k)$ and $z(\phi_k)$ denote the sum of weights of non-crossing edges based on our estimated edge weights and the exact edge weights, respectively. We have the following lemma to establish the convergence.

Lemma 2. $z'(\phi_k) \to z(\phi_k)$ as $\theta \to \infty$

Proof. By Theorem 1, we know that, for any edge $(u, v) \in E(G)$, when $\theta \to \infty$, we have $w_{(u,v)} \to \overline{w}_{(u,v)}$ where $\overline{w}_{(u,v)}$ is the exact weight value. Then the following holds:

$$\begin{aligned}z'(\phi_k) &= \sum_{\phi_k(u)=\phi_k(v),(u,v)\in E(G)} w_{(u,v)} \\ &\to \sum_{\phi_k(u)=\phi_k(v),(u,v)\in E(G)} \overline{w}_{(u,v)} = z(\phi_k)\end{aligned} \tag{4}$$

The lemma is thus deduced. □

In the next step, we analyze the error between $z'(\phi_k)$ and $z(\phi_k)$ w.r.t. the sample size θ.

Algorithm 1: SMGP+

Input : Original graph G, a query motif Q, the number of iteration t, a GP kernel M, the number of partitions k, the sample size θ.

Output: A partitioning plan ϕ_k.

1 **for** $j = 1$ *to* t **do**
2 $\quad w^j_{(\cdot,\cdot)} \leftarrow AdaptiveSampling(G, Q, \widetilde{\pi}, \theta)$;
3 $\quad$ **for** *each edge* $(u, v) \in E(G)$ **do**
4 $\quad\quad w^{avg}_{(u,v)} \leftarrow \frac{1}{j}\sum_{i=1}^{j} w^i_{(u,v)}$
5 $\quad \phi_k \leftarrow M(G, w^{avg}_{(\cdot,\cdot)}, k)$;
6 $\quad \widetilde{\pi} \leftarrow UpdateDistribution(G, w^{avg}_{(\cdot,\cdot)}, \phi_k, \widetilde{\pi})$;
7 **return** ϕ_k;

Lemma 3. Let OPT be the optimal objective value for $z(\cdot)$ among all partition plans. For a constant $\epsilon \in [0, 1]$, we have the following inequality $|z'(\phi_k) - z(\phi_k)| \leq \epsilon \cdot OPT$ hold for any partition plan ϕ_k with at least a probability of $1 - \delta$, when the sample size θ satisfies:

$$\theta \geq \frac{\eta^2 \log(\frac{1}{\delta})}{2\epsilon^2 OPT^2}$$

Proof. According to Theorem 1 and Equation 4, we have

$$\mathbb{P}[|z'(\phi_k) - z(\phi_k)| \geq \epsilon \cdot OPT] \leq \exp(-\frac{2\theta\epsilon^2 OPT^2}{\eta^2})$$

We can deduce the Lemma with $\delta = \exp(-\frac{2\theta\epsilon^2 OPT^2}{\eta^2})$. □

Based on Lemma 3, we proceed to prove the approximation ratio of SMGP.

Theorem 2. Given any partitioning algorithm that can achieve an approximation ratio of α $(0 < \alpha < 1)$ for maximizing the sum of weights of *non-crossing* edges based on the objective function $z(\cdot)$, applying this partitioning algorithm to SMGP can result in an $(\alpha - 2\epsilon)$-approximate solution with at least a probability of $1 - \delta$ when $\theta \geq \frac{\eta^2 \log(\frac{1}{\delta})}{2\epsilon^2 OPT^2}$.

Proof. Let ϕ^*_k denote the optimal partition plan on the graph with exact edge weights and $z(\phi^*_k) = OPT$. Assuming SMGP returns a partition plan ϕ_k with θ samples and let ϕ^s_k be the optimal partition plan on the graph with estimated edge weights. According to Lemma 3, we know that $|z'(\phi_k) - z(\phi_k)| \leq \epsilon \cdot OPT$ holds with at least a probability of $1 - \delta$ for all possible partition plans. Subsequently, we have:

$$\begin{aligned} z'(\phi_k) &\geq \alpha z'(\phi^s_k) \geq \alpha z'(\phi^*_k), \\ \text{and } z(\phi_k) &\geq z'(\phi_k) - \epsilon \cdot OPT \\ &\geq \alpha z'(\phi^s_k) - \epsilon \cdot OPT \\ &\geq \alpha z'(\phi^*_k) - \epsilon \cdot OPT. \qquad (5) \\ \text{Since } z'(\phi^*_k) &\geq z(\phi^*_k) - \epsilon \cdot OPT, \\ \text{then } z(\phi_k) &\geq \alpha(1-\epsilon)OPT - \epsilon \cdot OPT \\ &> (\alpha - 2\epsilon)OPT. \end{aligned}$$

Thus, the theorem is established. □

4. AN ADAPTIVE SAMPLING FRAMEWORK FOR EDGE WEIGHT ESTIMATION

While SMGP can deliver accurate partitioning results, it needs to sample an excessive number of subgraphs to ensure that the weight for each edge is estimated with a small error. For the MGP problem, we observe that not every simple edge plays a critical role in affecting the partitioning results and the samples dedicated to these non-significant edges are largely wasted. Motivated by this, we further propose a novel adaptive SMGP framework, namely SMGP+, which can dynamically adjust the sampling distribution of the starting edge to improve the partitioning quality while maintaining the same sample size.

4.1 An Overview of SMGP+

Algorithm 1 presents an overview of the SMGP+ framework which works by iteratively partitioning the original graph. In each iteration, we sample θ subgraphs according to a sampling distribution $\widetilde{\pi}$ for the starting edge (line 2). The estimated weights for iteration j, i.e., $w^j_{(\cdot,\cdot)}$, will be aggregated with the weights estimated from previous iterations to produce the up-to-date weight estimations $w^{avg}_{(\cdot,\cdot)}$ (line 4). The weight estimator $w^{avg}_{(u,v)}$ averaged across all t iterations will be close to its expected value $\mathbb{E}[w_{(u,v)}]$ with a small error as the number of iteration increases. We will show the convergence rate of $w^{avg}_{(u,v)}$ in Section 4.3. Subsequently, a GP kernel is invoked to partition G w.r.t. the weights $w^{avg}_{(\cdot,\cdot)}$ and generate a partitioning plan ϕ_k (line 5). Based on ϕ_k and $w^{avg}_{(\cdot,\cdot)}$, we will update the sampling distribution $\widetilde{\pi}$ to prioritize accurate weight estimation for those edges that could significantly affect the partitioning result. In this way, given a fixed sample size, we focus on sampling "important" edges to enable a better partitioning result. We will present our approach of updating the sampling distribution $\widetilde{\pi}$ in Section 4.2. Since these subgraphs are not sampled independently, we show the theoretical guarantee of SMGP+ by exploiting the concept of Martingale in Section 4.3.

4.2 Updating the Sampling Distribution

We update an existing sampling distribution $\widetilde{\pi}$ with the following equation:

$$\widetilde{\pi}((u, v)) \leftarrow \widetilde{\pi}((u, v)) \cdot \mathsf{PF}[(u, v)] \cdot \mathsf{NF}[(u, v)] \qquad (6)$$

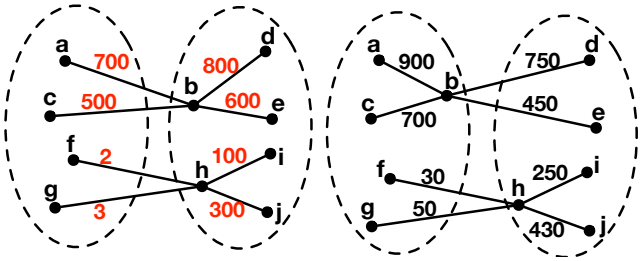
In Equation 6, we consider two factors in updating the sampling probability for (u, v): (1) the *Partition Factor* ($\mathsf{PF}[(u, v)]$) and (2) the *Neighborhood Factor* ($\mathsf{NF}[(u, v)]$) .

Partition Factor. We set a higher partition factor to the edge (u, v), if the inaccurate weight estimation of this edge can easily change the current partitioning plan on placing this edge. Let $\mathcal{C}(u, \varphi_i)$ denote the set of edges connecting the vertex u to vertices in the partition φ_i. If $i = \phi_k(u)$, then $\mathcal{C}(u, \varphi_i)$ contains all non-crossing edges incident on u. Otherwise, it contains all crossing edges connecting u to vertices in a different partition φ_i. Let $|\mathcal{C}(u, \varphi_i)|$ denote the sum of the weight of the edges in $\mathcal{C}(u, \varphi_i)$. The following equation measures how well u is connected to vertices in the same partition, as compared with those in a different partition:

$$\xi(u, \varphi_i) = |\mathcal{C}(u, \varphi_j)| - |\mathcal{C}(u, \varphi_i)|$$

where $j = \phi_k(u)$ and $i \neq j$.

We assume that the underlying GP kernel is effective such that $\xi(u, \varphi_i)$ is positive. If $\xi(u, \varphi_i)$ is negative, we can always move u to φ_i and reduce the sum of weights of the



(a) The optimal solution. (b) The solution of SMGP and SMGP+ with θ samples.

(c) SMGP with additional θ samples. (d) SMGP+ in the $2nd$ iteration with θ samples.

Figure 3: An example of using partition factor.

crossing edges. When $\xi(u, \varphi_i)$ is relatively small or close to zero, it is very likely that the inaccurate weight estimation of edges in $\mathcal{C}(u, \varphi_i)$ or $\mathcal{C}(u, \varphi_j)$ will make $\xi(u, \varphi_i)$ negative. If that is the case, the vertex u needs to be moved from φ_j to φ_i to reduce the edge-cut weight (i.e., to make $\xi(u, \varphi_i)$ positive and improve the partitioning objective value). The consequence caused by inaccurate weight estimations of such edges can potentially trigger a domino effect that changes the entire partitioning plan. On the contrary, if $\xi(u, \varphi_i)$ is estimated as a large positive value, then the exact value of $\xi(u, \varphi_i)$ is likely positive even when the weight estimations of involved edges are not accurate.

Example 4. Figure 3 shows an example on how the *partition factor* is used to improve the partitioning quality, where we only show a small part of the whole graph for easier illustration. Figure 3(a) refers to the optimal partitioning result based on the exact edge weights. When we set the sample size as θ, both SMGP and SMGP+ uniformly sample θ starting edges for weight estimation and may produce the partitioning result as shown in Figure 3(b). In this result, the sum of weights of edges (a, b) and (b, c) is close to the sum of weights of edges (b, d) and (b, e). The inaccurate weight estimation of these edges with high partition factors can easily influence the following decision: which partition the vertex b should be placed into, such that the edge-cut weight can be reduced? On the other hand, the sum of weights of (f, h) and (g, h) is much smaller than that of (h, i) and (h, j). More accurate weight estimations of these edges with low partition factors are unlikely to change the relationship between the sum of weights of these two pairs of edges. Thus, in the next iteration, the SMGP+ framework will assign (a, b), (b, c), (b, d) and (b, e) with higher sampling probabilities to compute more accurate weight estimations for them, and thereby produce a better partitioning result, as shown in Figure 3(d). On the contrary, SMGP still adopts the uniform sampling strategy and may fail to improve the partitioning quality even with a larger sample size as shown in Figure 3(c).

Motivated by the above intuition, for edges whose inaccurate weight estimations have great impacts on the partitioning result, we need to sample them as the starting edges with high probabilities, such that the partitioning result can be stabilized with more accurate edge weights. When we try to well-define the partition factor, several aspects need to be considered:

1. For a crossing edge (u, v) where $\phi_k(u) = j, \phi_k(v) = i$, the values of $\xi(u, \varphi_i)$ and $\xi(v, \varphi_j)$ are different. It indicates that the inaccurate weight estimation for (u, v) has different levels of impact on movements of vertices u and v. Thus, the partition factor of (u, v) is different when analyzing vertex u and vertex v individually. Since we need to make sure the weight estimation of (u, v) is accurate enough such that neither of these two vertices needs to be moved, a larger partition factor between u and v should be used as the final partition factor of (u, v) (Equation 7).

2. Similarly, for a non-crossing edge (u, v) where $\phi_k(u) = \phi_k(v) = j$, its estimated weight is involved in every pair of $\xi(u, \varphi_i)$ $(1 \leq i \leq k, i \neq j)$. That means the inaccurate estimation for (u, v) has different influence on which partition φ_i we should move u into. Thus, we also consider the largest $\xi(u, \cdot)$ among all pairs to compute the partition factor of a non-crossing edge (u, v) (Equation 8).

Based on the above aspects, we devise the partition factor $\mathsf{PF}[(u, v)]$ of an edge (u, v) as follows:

$$\mathsf{PF}[(u, v)] = max(\mathsf{PF}_u[(u, v)], \mathsf{PF}_v[(u, v)]) \tag{7}$$

where $\mathsf{PF}_u[(u, v)]$ and $\mathsf{PF}_v[(u, v)]$ are defined symmetrically:

$$\mathsf{PF}_u[(u, v)] = \begin{cases} \exp(-\frac{\xi(u, \varphi_i)}{d_u} + 1), & \text{if } \phi_k(u) \neq \phi_k(v) = i \\ \exp(-\frac{\xi_{min}(u)}{d_u} + 1), & \text{otherwise} \end{cases}$$

and

$$\xi_{min}(u) = \min_{\phi_k(u)=j, 1 \leq i \leq k, i \neq j} |\mathcal{C}(u, \varphi_j)| - |\mathcal{C}(u, \varphi_i)| \tag{8}$$

Neighborhood Factor. It is worth mentioning that there could be a notably higher number of motif instances discovered in the dense region of the graph than those discovered in the sparse region. The different number of motif instances results in weight difference, and higher weights have a significant impact on the partitioning result. An edge may appear in either a sparse or a dense region of the graph. Hence we penalize an edge e if it appears in a sparse region of the graph and lower the sampling probability. To achieve this goal, we look at the neighborhood of an edge $e = (u, v)$ and use the minimum unweighted degree of the endpoints of e to indicate the 'density' of the neighborhood, i.e., $min(|N_u|, |N_v|)$. Additionally, it is possible that the neighbor size of high-degree nodes is significantly larger than that of the remaining nodes in power-law graphs. To ensure that each edge is assigned

with a reasonable sampling probability and outliers are penalized, we apply the log function in $\mathsf{nf}[(\mathsf{u},\mathsf{v})]$ as follows:

$$\mathsf{nf}[(\mathsf{u},\mathsf{v})] = \log min(|N_u|, |N_v|)$$

Sampling Distribution Computation. Note that computing the sampling probability of each edge requires traversing the whole graph, which is prohibitively costly. To speedup the sampling distribution computation, we first sample a pool of vertices (e.g., 100,000) based on their weighted degrees and then update the sampling distribution of edges connecting these vertices. In this way, we can avoid processing the entire graph and our experiments have demonstrated that this process is efficient and still obtains high-quality partitioning result compared with the one using exact edge weights.

4.3 Theoretical Analysis

In contrast to the theoretical analysis of SMGP (in Section 3.2), the samples in SMGP+ are not independently generated due to the adaptive sampling strategy devised. As a result, it makes the Hoeffdding inequality inapplicable to show how the edge weights estimated by SMGP+ converge to the exact weights. Thus, we propose to employ the martingale, a classical statistic tool, to restore the theoretical guarantee of SMGP+.

Definition 7. Martingale [15]. A sequence of random variables $Z_0, Z_1, ..., Z_t$ (possibly correlated) is a martingale with respect to the sequence $X_0, X_1, ..., X_t$ if the following conditions hold for all $0 \leq i \leq t$:

1. Z_i is a function of $X_0, X_1, ..., X_i$;
2. $\mathbb{E}[|Z_i|] < \infty$;
3. $\mathbb{E}[Z_{i+1}|Z_1, Z_2, ..., Z_i] = Z_i$

In SMGP+, we iteratively update the weight estimation $w^i_{(u,v)}$ where $1 \leq i \leq t$. To connect $w^i_{(u,v)}$ to martingales, we have

$$w_{(u,v)} = \frac{1}{t}\mathbb{E}[\sum_{i=1}^{t} w^i_{(u,v)}] \tag{9}$$

Furthermore, only the starting edge's sampling distributions $\widetilde{\pi}_1, \widetilde{\pi}_2, ..., \widetilde{\pi}_t$ in each iteration are correlated. Give the query motif Q, $\widetilde{\pi}_i (1 \leq i \leq t)$ is supported over all edges and thus unbiased, and the subgraphs of size $|V(G)|$ expanded from the starting edges are sampled uniformly. Due to these properties, for any $i \in [1, t]$, we have

$$\mathbb{E}[w^i_{(u,v)}|w^1_{(u,v)}, w^2_{(u,v)}, ..., w^{i-1}_{(u,v)}] = \mathbb{E}[w^i_{(u,v)}] = w_{(u,v)}$$

Let $R_i = \sum_{j=1}^{i}(w^j_{(u,v)} - w_{(u,v)})$, we have $\mathbb{E}[R_i] = 0$ and

$$\mathbb{E}[R_i|R_1, R_2, ..., R_{i-1}] = R_{i-1}$$

Therefore, by Definition 7, $R_1, R_2, ..., R_t$ is a martingale.

The following lemma shows two concentration results for martingales if some properties of martingales are known.

Lemma 4. ([15]). Let $Z_1, Z_2, ..., Z_t$ be a martingale, such that $|Z_1| \leq a$, $|Z_i - Z_{i-1}| \leq a$ $(2 \leq i \leq t)$, and

$$Var[Z_1] + \sum_{i=2}^{t} Var[Z_i|Z_1, Z_2, ..., Z_{i-1}] \leq b,$$

where $Var[\cdot]$ refers to the variance of a random variable. Then for any β,

$$\mathbb{P}(Z_i - \mathbb{E}[Z_i] \geq \beta) \leq \exp(-\frac{\beta^2}{\frac{2}{3}a\beta + 2b}).$$

Recall the definition of R_i, we can prove that our martingale $R_1, R_2, ..., R_h$ have the above properties to reach concentration results by equivalently showing: $|w^i_{(u,v)}|$ can be bounded by the same constant and the (conditional) variance of each $w^i_{(u,v)}$ can be bounded as well.

Based on our analysis in Section 3, the lower-bound of each $w^i_{(u,v)}$ is 0 and we can find the upper-bound for each $w^i_{(u,v)}$ as η. Thus, we have $|R_1| < \eta$ and $|R_i - R_{i-1}| < \eta$ $(2 \leq i \leq t)$ due to the definition of R_i. This in turn derives the following:

$$Var[R_1] + \sum_{i=2}^{t} Var[R_i|R_1, R_2, ..., R_{i-1}] = \sum_{i=1}^{t} Var[R_i] < t\eta^2,$$

As a result, based on Lemma 4, we can have the following concentration results.

Theorem 3. Given the edge weights $w^1_{(u,v)}, w^2_{(u,v)}, ..., w^t_{(u,v)}$ computed by our SMGP+ framework in t iterations based on the query motif Q, and the maximum possible estimated edge weight η, the deviation of the averaged edge weight of (u, v) from $\mathbb{E}[w_{(u,v)}]$ is bounded with a probability increased with t, such that, for any $\epsilon > 0$,

$$\mathbb{P}[\frac{1}{t}\sum_{i=1}^{t} w^i_{(u,v)} - \mathbb{E}[w_{(u,v)}] \geq \epsilon] \leq \exp(-\frac{\epsilon^2}{\frac{2}{3}\epsilon\eta + 2\eta^2}t),$$

$$\mathbb{P}[\frac{1}{t}\sum_{i=1}^{t} w^i_{(u,v)} - \mathbb{E}[w_{(u,v)}] \leq -\epsilon] \leq \exp(-\frac{\epsilon^2}{\frac{2}{3}\epsilon\eta + 2\eta^2}t).$$

Proof. Given Lemma 4, $R_i = \sum_{j=1}^{i}(w^j_{(u,v)} - w_{(u,v)})$ and $\mathbb{E}[R_i] = 0$, for any $\epsilon > 0$, we have

$$\mathbb{P}[\sum_{i=1}^{t} w^i_{(u,v)} - t\mathbb{E}[w_{(u,v)}] \geq \epsilon t] \leq \exp(-\frac{\epsilon^2 t^2}{\frac{2}{3}\epsilon t\eta + 2t\eta^2})$$

On the other hand, by applying Lemma 4 on the martingale $-R_1, -R_2, ..., -R_t$, we have

$$\mathbb{P}[\sum_{i=1}^{t} w^i_{(u,v)} - t\mathbb{E}[w_{(u,v)}] \leq -\epsilon t] \leq \exp(-\frac{\epsilon^2 t^2}{\frac{2}{3}\epsilon t\eta + 2t\eta^2})$$

Thus, this theorem is deduced. □

By following the arguments made in Lemmas 2-3 and Theorem 2 for SMGP, Theorem 2 naturally leads to the approximation guarantee of SMGP+ in Theorem 4. Thus, we omit the proof.

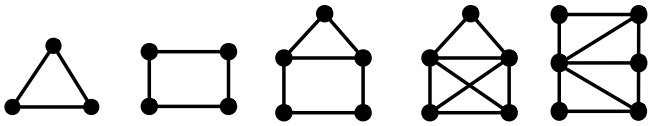
Theorem 4. Given any α-approximate partitioning algorithm for maximizing the weighted sum of non-crossing edges defined based on the objective function $z(\cdot)((0 < \alpha < 1)$, applying this partitioning algorithm to SMGP+ can result in an $(\alpha - 2\epsilon)$-approximate solution with at least $1 - \delta$ probability when $t \geq \frac{(\frac{2}{3}\epsilon\eta + 2\eta^2)\log(\frac{1}{\delta})}{\epsilon^2 OPT^2}$.

Datasets	Nodes	Edges	Average Degree
Hyves	1,402,673	2,777,419	4.0
Flixster	2,523,386	7,918,801	6.3
Gowalla	196,591	950,327	9.6
LiveJournal	5,204,176	49,174,464	18.9
Orkut	3,072,441	117,184,899	76.3

Table 2: The statistics of datasets

Dataset	Statistics	Query Motif				
		Q_1	Q_2	Q_3	Q_4	Q_5
Hyves	**Time (s)**	5.0E+1	5.0E+2	5.0E+2	5.0E+2	3.6E+4
	Count	1.5E+6	4.2E+7	1.2E+9	2.9E+8	1.9E+10
Flixster	**Time (s)**	1.0E+2	3.9E+3	1.1E+5	5.3E+4	8.0E+5
	Count	1.6E+7	3.1E+8	8.9E+9	5.0E+9	1.2E+11
Gowalla	**Time (s)**	5.0E+1	8.0E+2	7.7E+4	2.3E+4	2.7E+6
	Count	4.6E+6	8.5E+7	6.6E+9	2.3E+9	2.2E+11
LiveJournal	**Time (s)**	1.3E+3	7.0E+4			
	Count	6.2E+8	1.1E+10			
Orkut	**Time (s)**	2.6E+3	8.1E+5			
	Count	1.3E+9	1.4E+11			

Table 3: Statistics of EMGP.



(a) Q_1 (b) Q_2 (c) Q_3 (d) Q_4 (e) Q_5

Figure 4: Query motifs used in the experiments.

5. EXPERIMENT

In this section, we conduct experiments over several real-world datasets to compare our proposed edge estimation approaches, i.e., SMGP and SMGP+, with the exact MGP method. We compare their performance based on the partitioning results produced by three representative GP kernels with different popular partition quality evaluation metrics).

Datasets. Five real-world datasets, i.e., Hyves, Flixster, Gowalla, LiveJournal and Orkut[1], are used. Table 2 summarizes the characteristics of each dataset.

Query Motifs. We test five different motifs as shown in Figure 4. These five motifs are frequent patterns in our test datasets and are relatively easy to be computed for exact weights due to their symmetric property. Results on more motifs that cover symmetric and asymmetric graph patterns are available at our technical report [1], and we hereby do not include them due to space limit.

Methods for comparison. We compare the following weight estimation methods for the MGP problem.

- Exact MGP (EMGP) [41]: a state-of-the-art method for exact motif counting. It reduces redundant computations in DFS by deferring the materialization of pattern vertices until necessary and converting the candidate set computation into the problem of finding a minimum set cover. The exact motif count and the running time of EMGP are shown in Table 3. Results that cannot be finished within a month are not reported. The results on Livejournal and Orkut are available at our technical report [1].

[1]Dataset are collected from http://konect.uni-koblenz.de

- SMGP: It uniformly samples starting edges to expand subgraphs for edge weight estimations, and then applies the underlying GP kernel to produce the result based on the estimated weights.
- SMGP+: In each iteration, it adaptively samples starting edges for edge weight estimations based on the partitioning result in the last iteration. Then the underlying GP kernel is applied to refine the partitioning result based on the updated edge weights.

Underlying GP kernels. To showcase that our proposed methods are general to work with different choices of GP kernels and objective functions for MGP, we evaluate the performance of baselines by using three widely-adopted GP kernels with different evaluation metrics (i.e., edge-cut weight [28], modularity [42] and conductance [3]).

- `Metis` [28]: it aims to produce balanced partitioning results with a minimum *edge-cut weight* by collapsing vertices and edges, partitioning the smaller graph, and then uncoarsening it to construct a partition for the original graph.
- `MAPPR` [46]: it aims to compute the partitioning results with the minimum *conductance*, by sweeping over a motif-based approximate personalized PageRank vector of vertices.
- `Fennel` [42]: it formulates an objective function that generalizes the problem of maximizing the *modularity* scores by unifying two popular partitioning heuristics: placing the newly arrived vertex into (1) the cluster with the largest number of neighbors, or (2) the cluster with the least number of non-neighbors. In our case, we adjust these heuristics based on the sum of the weights of the edges connecting the newly arrived vertex to these neighbors.

Environments. We conduct all experiments on a Linux server with Intel Xeon E5 (2.60 GHz) CPUs and 512 GB RAM. All codes are implemented in Python and we use a state-of-the-art method [41] for the exact edge weight computation. For a fair comparison, all algorithms are executed with a single thread.

Parameter settings for estimation methods. In each iteration, we set the sample size $\theta = |V(Q)|^2 \cdot |E(G)| \cdot r$ for SMGP+, where Q is the query motif, G is the dataset and r is a sample ratio. When the underlying GP kernel is `Metis`, we set $r = 0.02$. When the underlying GP kernel is `Fennel` or `MAPPR`, we set $r = 0.01$. To better compare SMGP+ and SMGP, we also vary the sample size θ of SMGP and compare the partitioning quality of SMGP and SMGP+ with different number of samples. Note that, for SMGP, the partitioning process in each iteration for quality evaluation will not contribute to the running time cost of SMGP.

Parameter settings for GP kernels. We set the number of partitions $k = 10$ for `Fennel` and `Metis`. We set the load threshold parameter of `Fennel` as 5. Since `MAPPR` is a local clustering method, we follow the common standard to set $k = 2$ for `MAPPR`. For all other parameters of these partitioning methods, we adopt the default setting as reported in the original papers.

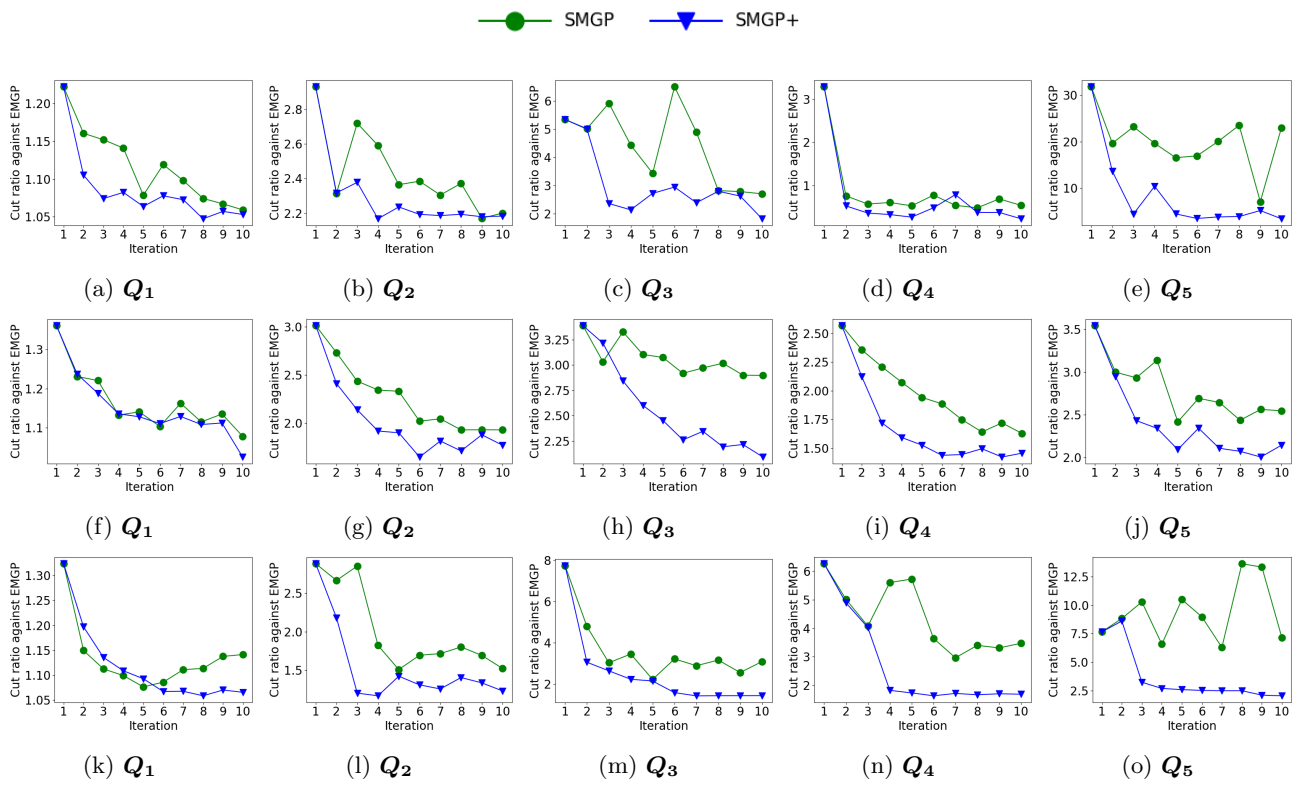


Figure 5: Performance on Hyves (1*st* row), Flixster (2*nd* row) and Gowalla (3*rd* row) with `Metis` as the GP kernel.

5.1 Effectiveness Evaluation

When evaluating the quality of the partitioning results produced by EMGP, SMGP and SMGP+, we adopt the evaluation metric corresponding to the objective function of the underlying GP kernel as shown in Table 4.

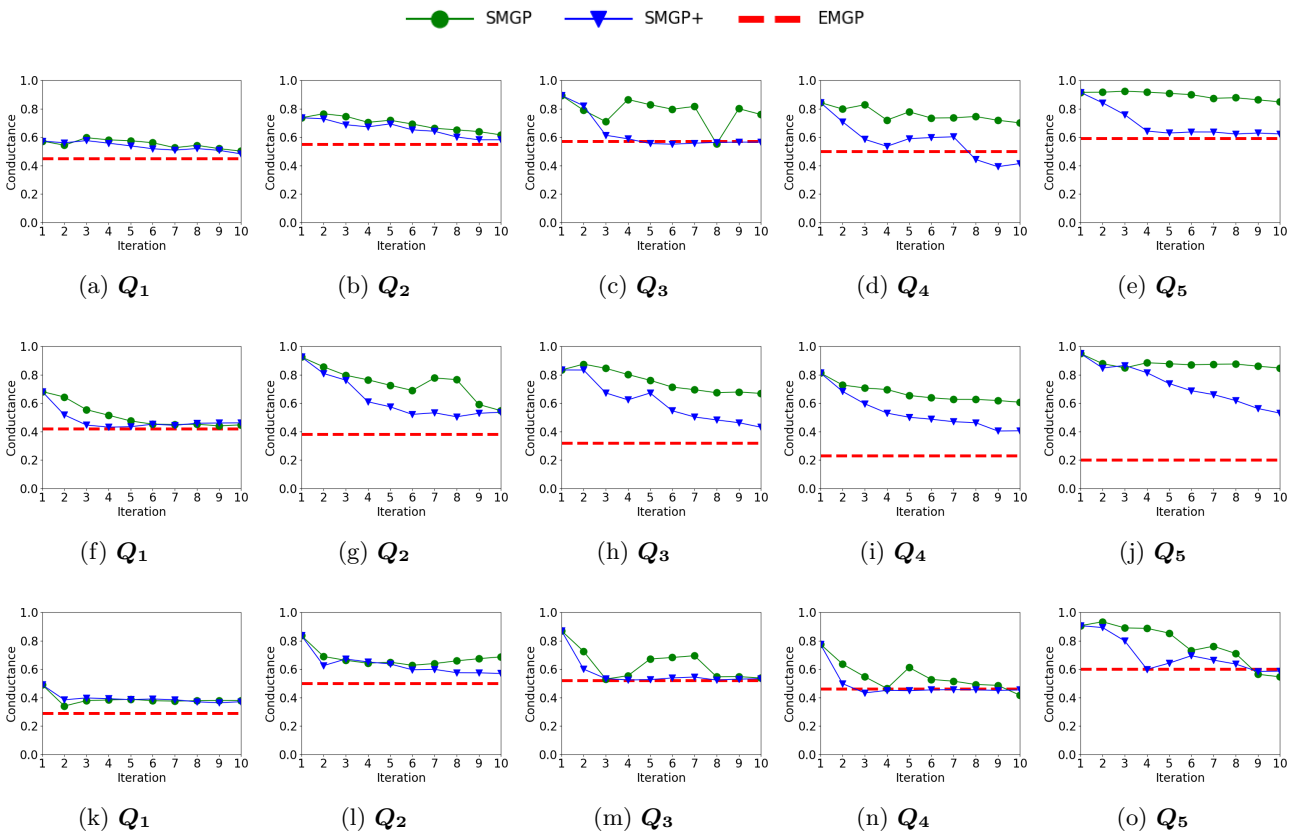
GP kernels	Objective Function	Evaluation Metric
Metis	Minimizing the edge-cut weight	Cut ratio
MAPPR	Minimizing the conductance	Conductance
Fennel	Maximizing the modularity	Modularity

Table 4: Evaluation metrics for underlying GP kernels.

Edge-cut weight evaluation with Metis: Instead of individually evaluating the edge-cut weight of different methods for edge weight estimation, we are more interested in how close the quality of results produced by SMGP or SMGP+ is to that of EMGP. Formally, we evaluate the performance of SMGP and SMGP+ based on the cut ratio $\frac{Cut(\phi'_k)}{Cut(\phi_k)}$, where the partition plan ϕ'_k is produced with edge weights estimated by either SMGP or SMGP+, ϕ_k is produced with the exact edge weights computed by EMGP, and $Cut(\cdot)$ calculates the sum of the exact edge weights of those crossing edges in a partitioning plan. Thereby, the lower the cut ratio is, the better quality the partitioning plan achieves. If the cut ratio is close to 1, the quality of partitioning plan produced by SMGP or SMGP+ is close to the one produced by EMGP. Figure 5 shows the cut ratio achieved by SMGP and SMGP+ with `Metis` as the underlying GP kernel. As the number of iteration increases, the cut ratio achieved by SMGP+ is generally getting better and, in some cases, can be very close to 1 (e.g., Figure 5(a),(d),(f),(k) and (l)). It thereby indicates the effectiveness of the adaptive sampling strategy in SMGP+. Also, SMGP is as competitive as SMGP+ in some scenarios (e.g., Figure 5(a) and (d)) but its performance is generally unstable and outperformed by SMGP+. This is mainly because SMGP fails to identify those edges whose inaccurate weight estimations may bring notable impacts on the partitioning result. Hence the weight estimations for these important edges may be still inaccurate for more samples, and such inaccuracies can trigger domino effects that change the entire partitioning plan.

Conductance evaluation with MAPPR. Figure 6 compares the conductance achieved by different weight estimation methods with `MAPPR` as the GP kernel. The partitioning quality produced by SMGP can be competitive with the one produced by EMGP based on the exact edge weights in some cases (e.g., Figure 6(a) and (f)). On the other hand, in many cases, the conductance achieved by SMGP decreases quite slowly as the number of iteration increases. In contrast, SMGP+ is able to produce a much better partitioning quality than SMGP given the same sample size, and achieves a nearly-identical partitioning quality compared with EMGP in many cases (e.g., Figure 6(c), (e) and (n)).

Modularity evaluation with Fennel: Figure 7 shows the modularity achieved by SMGP and SMGP+ with `Fennel` as the graph kernel. The modularity achieved by SMGP+ increases notably faster than that of SMGP as the iteration grows, which again shows the significance of the adaptive sampling strategy. On the sparsest dataset Hyves, SMGP+ is able to achieve very competitive modularity scores compared with EMGP (e.g., Figure 7(c),(d) and (e)). On the other hand, SMGP+ can produce good partitioning results with motif Q_1 and Q_4 on relatively denser datasets Flixster and Gowalla, but its performance with the query motifs Q_2,

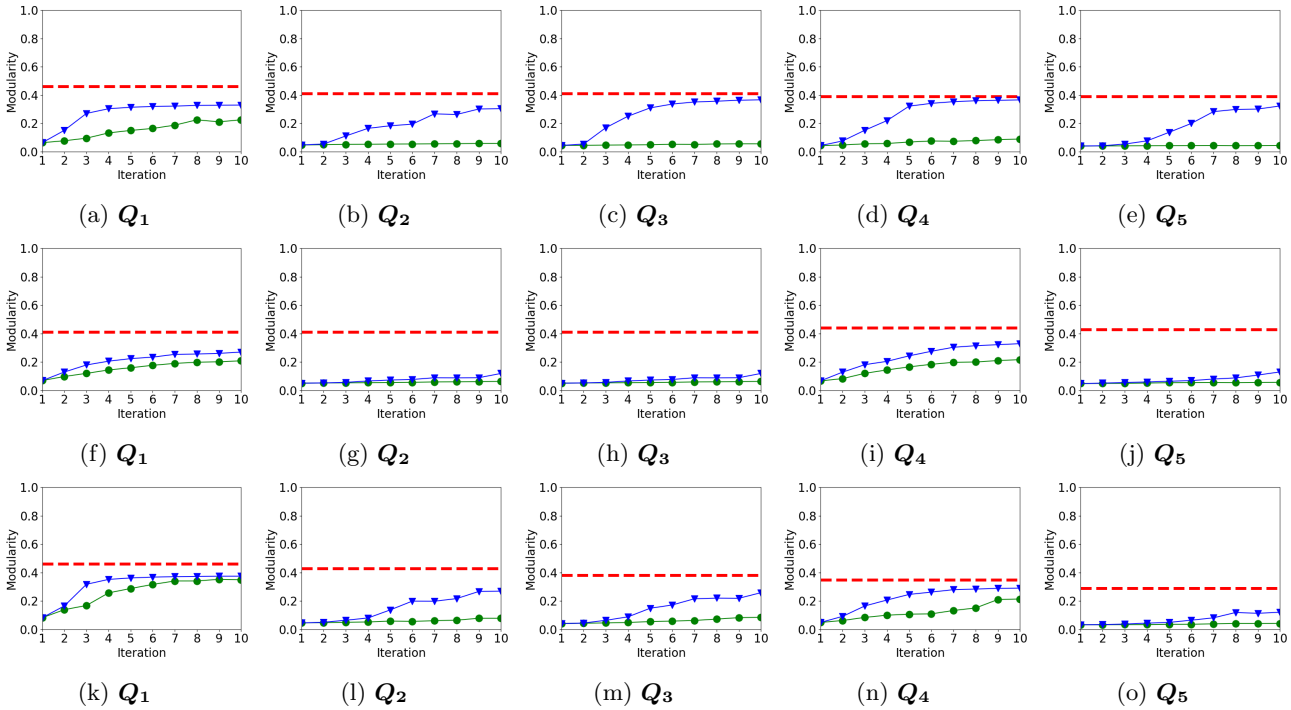


(a) $\boldsymbol{Q_1}$ (b) $\boldsymbol{Q_2}$ (c) $\boldsymbol{Q_3}$ (d) $\boldsymbol{Q_4}$ (e) $\boldsymbol{Q_5}$

(f) $\boldsymbol{Q_1}$ (g) $\boldsymbol{Q_2}$ (h) $\boldsymbol{Q_3}$ (i) $\boldsymbol{Q_4}$ (j) $\boldsymbol{Q_5}$

(k) $\boldsymbol{Q_1}$ (l) $\boldsymbol{Q_2}$ (m) $\boldsymbol{Q_3}$ (n) $\boldsymbol{Q_4}$ (o) $\boldsymbol{Q_5}$

Figure 6: Performance on Hyves (1*st* row), Flixster (2*nd* row) and Gowalla (3*rd* row) with `MAPPR` as the GP kernel.

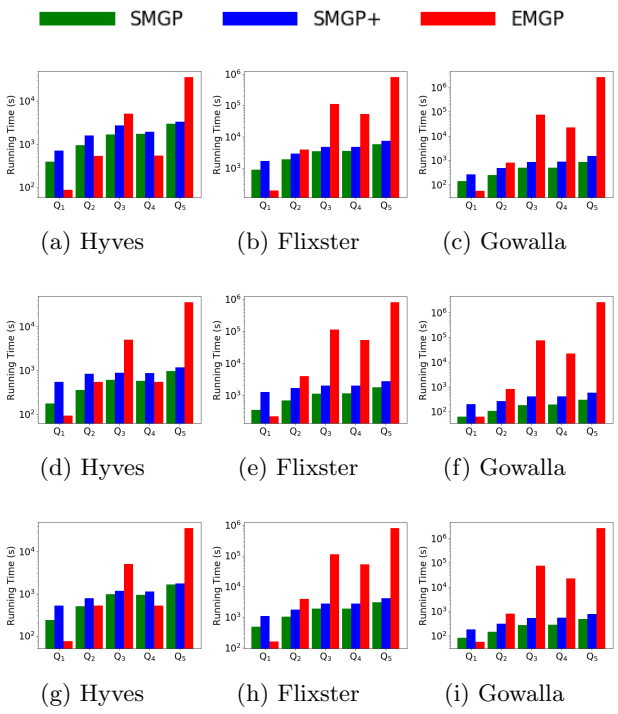


(a) $\boldsymbol{Q_1}$ (b) $\boldsymbol{Q_2}$ (c) $\boldsymbol{Q_3}$ (d) $\boldsymbol{Q_4}$ (e) $\boldsymbol{Q_5}$

(f) $\boldsymbol{Q_1}$ (g) $\boldsymbol{Q_2}$ (h) $\boldsymbol{Q_3}$ (i) $\boldsymbol{Q_4}$ (j) $\boldsymbol{Q_5}$

(k) $\boldsymbol{Q_1}$ (l) $\boldsymbol{Q_2}$ (m) $\boldsymbol{Q_3}$ (n) $\boldsymbol{Q_4}$ (o) $\boldsymbol{Q_5}$

Figure 7: Performance on Hyves (1*st* row), Flixster (2*nd* row) and Gowalla (3*rd* row) with `Fennel` as the GP kernel.



(a) Hyves (b) Flixster (c) Gowalla

(d) Hyves (e) Flixster (f) Gowalla

(g) Hyves (h) Flixster (i) Gowalla

Figure 8: Efficiency comparison with `Metis` (1*st* row), `MAPPR` (2*nd* row) and `Fennel` (3*rd* row) as the graph kernels.

Q_3 and Q_5 is less satisfying. As shown in Table 3, the number of instances of Q_2, Q_3 and Q_5 is notably larger than the one of Q_1 and Q_4 in Gowalla and Flixster, which indicates that the edge weight estimation with the former three motifs may be less accurate given the same number of samples. In spite of this fact, we can still see that SMGP+ shows promising performance with other GP kernels based on these three motifs (as shown in Figure 5 and Figure 6), which indicates that `Fennel` is more sensitive to the inaccurate weight estimation and thus more samples are needed.

To summarize, our methods can produce high-quality results with very small sample sizes in many cases. We think that the performance of our methods can be boosted with large sample sizes, which will be demonstrated in the parameter analysis part.

5.2 Efficiency Comparison

Figure 8 compares the efficiency of SMGP, SMGP+ and EMGP. Considering that the exact edge weight computation is very costly, we select three relatively sparse real-world datasets for experiments. We have the following findings:

1. In spite of the low density of these datasets and very small sample sizes, SMGP+ can be four orders of magnitude faster than EMGP while achieving competitive partitioning quality (e.g., comparing Figure 6(o) and the last grouped bars in Figure 8(i)).

2. From these three figures and Table 3, we can see EMGP is very sensitive to the density of datasets and the query motifs, and its running time is mostly dominated by the exact edge weight computations. For instance, with Q_3 as the query motif, EMGP spends two orders of magnitude more time on computing the exact edge weights on Gowalla than it spends on Hyves. For another example, in the Gowalla dataset, EMGP needs to spend five orders of magnitude more time on computing the exact weights for Q_5 than it does for Q_1.

3. Compared with EMGP, the performance of SMGP and SMGP+ is very stable across different motifs.

4. EMGP runs faster than SMGP and SMGP+ with some simple query motifs (e.g., Q_1 and Q_2) but its efficiency decays as the density and scale of the dataset increases. Exact counting on simple motifs normally can be speeded up with specialized domain knowledge. For example, triangle counting, as a special case of motif counting [34], can be computed efficiently for most graphs, even for the graphs with a skewed degree distribution [7]. However, the exact computation loses this advantage against more complicated graph patterns which are also important for the MGP problem [6].

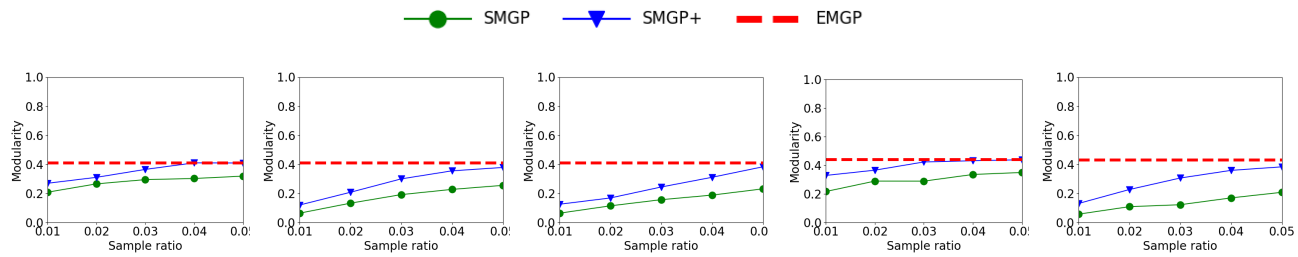
5.3 Parameter Analysis

There are two major parameters associated with our methods, i.e., the iteration number t and the sample size θ. Since we have analyzed the influence of t on the performance with a fixed θ earlier, we hereby focus on analyzing how the performance of our methods perform w.r.t. different sample sizes. By comparing all previous results, we find there is still a notable gap for SMGP and SMGP+ to improve their performance on Flixster with `Fennel` as the GP kernel, because the modularity scores achieved are low compared with EMGP (e.g., Figure 7(j)). Thus, we decide to analyze how their performance can be boosted in this case when the sample size increases. Figure 9 to Figure 10 compare the effectiveness and efficiency of SMGP and SMGP+ with $t = 10$ but different sample ratios. As the sample ratio increases, the effectiveness of SMGP and SMGP+ is boosted and SMGP+ can eventually produce results whose quality is nearly-identical to the one produced by EMGP in many cases. Even if the running time also increases with greater sample ratios, SMGP+ can still obtain up to one order of magnitude speedup while achieving nearly-identical partitioning quality compared to EMGP. Although SMGP+ is slower than SMGP, SMGP+ is clearly a better choice as it can generally produce better partitioning results than the latter one given a less amount of time. For example, when we assign SMGP and SMGP+ with a sample ratio 0.05 and 0.02 respectively, SMGP+ is able to spend only half of time spent by SMGP to achieve a greater modularity score as shown in Figure 9(e) and Figure 10(e).

6. RELATED WORK

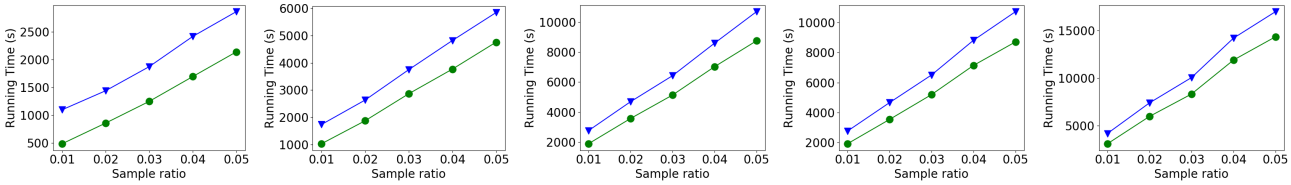
We summarize the literature related to this work from two fields, namely graph partitioning and graphlet computation.

Graph Partitioning is a fundamental problem in graph theory that has wide applications in community detection [19], distributed computing and image processing [4]. *edge-cut weight* [29], *conductance* [3] and *modularity* [16] are three popular metrics to evaluate the partition quality. Given a connected graph G, finding a bisection with the minimum conductance [44] or the maximum modularity [9] is NP-hard. In addition to the NP-hardness, many practical requirements of graph partition makes the problem even more intricate. For example, the problem of balanced k-way graph



(a) Q_1 (b) Q_2 (c) Q_3 (d) Q_4 (e) Q_5

Figure 9: Effectiveness comparison based on different sample ratios on Flixster with `Fennel` as GP.



(a) Q_1 (b) Q_2 (c) Q_3 (d) Q_4 (e) Q_5

Figure 10: Efficiency comparison based on different sample ratios on Flixster with `Fennel` as GP.

partition with simple minimum edge cut is proven to be NP-hard as well [4]. As such, there are a long line of researches aiming to develop efficient k-way partition algorithms that can achieve good empirical results and easy-to-adjust optimization objectives [31, 18, 27, 40, 42].

The aforementioned existing studies focus on analyzing only simple edges across the partitions and designing optimization algorithms correspondingly. Recently, it has been shown that high-order connected subgraphs, known as network motifs, can provide more insights to understand fundamental graph structures and behaviors of many complex systems [14, 25, 13, 46, 39]. This finding drives a surge in developing methods for motif-based graph partitioning [46, 6, 5, 33, 38, 43]. Most of these methods try to generalize simple edge conductance to motif conductance, and develop high order spectral graph partitioning algorithms such that the returned partitions have low motif conductance. Unfortunately, existing motif-based partition methods have two drawbacks. First, they only focus on the conductance metric while it is not straightforward to extend these solutions to handle more requirements raised by varying applications in reality, e.g., balanced k-way partition offered by most graph partitioning libraries [27, 11, 29]. Second, they have to compute the *entire motif adjacency matrix* by invoking existing algorithms on graphlet computation to count the number of times two nodes co-occur in an instance of the motif, and then apply subsequent methods to the matrix for partitioning. Computing the motif adjacency matrix is computationally expensive in nature, especially for large scale graphs and complex motifs with more than 3 vertices. To mitigate these issues, we propose an adaptive sampling framework to estimate edge weights, and our framework is robust to the choice of partitioning objectives (e.g., edge-cut weight , conductance and modularity) to cater for different application needs, as demonstrated in the experiments.

Graphlet Computation is a closely related field to motif analysis. Given a data graph G and a connected motif graph Q, graphlet count is the number of induced sub-graphs on G that are isomorphic to Q, which is computationally intensive since determining whether G contains a subgraph that is isomorphic to Q is NP-complete [17]. Due to the high computational cost of exact solutions [12, 21, 32, 2, 34], many sampling-based techniques are proposed. Monte Carlo based methods are shown to be very successful in accurately estimating the graphlet count with guaranteed high probabilities [8, 30, 45, 10]. In [30], a random walk method is proposed to estimate the 3-vertex graphlet count so as to analyze the clustering coefficient of social networks. To improve upon prior work, the authors in [45, 10] generate statistics of graphlet count on high order graphs. Paramonov et al. [36] propose a new Monte Carlo algorithm, called lifting, for graphlet count computation. The aforementioned solutions lay the foundation of the sampling approach in this work. In particular, we propose a novel edge-centric sampling method in contrast to node-centric sampling adopted by existing studies for graphlet computation. The edge-centric method allows us to develop the adaptive sampling framework to achieve efficient MGP computations.

7. CONCLUSION

In this paper, we study how to efficiently estimate motif-based edge weights for the motif-based partitioning problem (MGP). Existing MGP methods suffer from the prohibitively expensive cost of computing exact motif-based edge weights which are then fed to the underlying graph partitioning (GP) kernel to produce the partitioning result. To mitigate this issue, we first propose a sampling-based MGP method which is an unbiased estimator of exact edge weights and enable the GP kernel to produce some promising results based on the estimated edge weights. To further improve the efficiency, we propose a novel adaptive sampling framework called SMGP+. It iteratively performs partitioning with the underlying GP kernel based on up-to-date estimated edge weights, and adaptively adjusts the sampling distribution so that edges that are more likely to affect the partitioning outcome will be prioritized for weight estimation. Extensive experiments have demonstrated the effectiveness and efficiency of our approaches.

8. REFERENCES

[1] `http://shixunh.io/TechnicalReport.pdf`.

[2] F. N. Afrati, D. Fotakis, and J. D. Ullman. Enumerating subgraph instances using map-reduce. In *ICDE*, pages 62–73, 2013.

[3] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, pages 475–486, 2006.

[4] K. Andreev and H. Racke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.

[5] A. R. Benson, D. F. Gleich, and J. Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *SDM*, pages 118–126, 2015.

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

[7] J. W. Berry, L. K. Fostvedt, D. J. Nordman, C. A. Phillips, C. Seshadhri, and A. G. Wilson. Why do simple algorithms for triangle enumeration work in the real world? In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 225–234, 2014.

[8] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *ICDM*, pages 91–100, 2012.

[9] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 121–132, 2007.

[10] X. Chen, Y. Li, P. Wang, and J. Lui. A general framework for estimating graphlet statistics via random walk. *VLDB*, 10(3):253–264, 2016.

[11] C. Chevalier and F. Pellegrini. Pt-scotch: A tool for efficient parallel graph ordering. *Parallel computing*, 34(6-8):318–331, 2008.

[12] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985.

[13] F. Chierichetti, R. Kumar, P. Raghavan, and T. Sarlos. Are web users really markovian? In *WWW*, pages 609–618, 2012.

[14] K.-K. R. Choo. Money laundering risks of prepaid stored value cards. *Trends & Issues in Crime & Criminal Justice*, (363), 2008.

[15] F. Chung and L. Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79–127, 2006.

[16] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[17] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[18] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pages 175–181, 1982.

[19] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[20] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 28(3):469–475, 2006.

[21] J. A. Grochow and M. Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106, 2007.

[22] G. Han and H. Sethu. Waddling random walk: Fast and accurate mining of motif statistics in large graphs. In *ICDM*, pages 181–190, 2016.

[23] B. Hendrickson and T. G. Kolda. Graph partitioning models for parallel computing. *Parallel computing*, 26(12):1519–1534, 2000.

[24] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. 1994.

[25] C. J. Hoofnagle. Identity theft: Making the known unknowns known. *Harv. JL & Tech.*, 21:97, 2007.

[26] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

[27] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[28] G. Karypis and V. Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[29] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI design*, 11(3):285–300, 2000.

[30] L. Katzir and S. J. Hardiman. Estimating clustering coefficients and size of social networks via random walk. *TWEB*, 9(4):19, 2015.

[31] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[32] H. Kim, J. Lee, S. S. Bhowmick, W.-S. Han, J. Lee, S. Ko, and M. H. Jarrah. Dualsim: Parallel subgraph enumeration in a massive graph on a single machine. In *SIGMOD*, pages 1231–1245, 2016.

[33] C. Klymko, D. Gleich, and T. G. Kolda. Using triangles to improve community detection in directed networks. *arXiv preprint arXiv:1404.5874*, 2014.

[34] L. Lai, L. Qin, X. Lin, and L. Chang. Scalable subgraph enumeration in mapreduce. *VLDB*, 8(10):974–985, 2015.

[35] X. Lu and S. Bressan. Sampling connected induced subgraphs uniformly at random. In *International Conference on Scientific and Statistical Database Management*, pages 195–212, 2012.

[36] K. Paramonov, D. Shemetov, and J. Sharpnack. Estimating graphlet statistics via lifting. In *KDD*, pages 587–595, 2019.

[37] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[38] K. Rohe and T. Qin. The blessing of transitivity in sparse and stochastic networks. *arXiv preprint*

arXiv:1307.2302, 2013.

[39] O. Sporns and R. Kötter. Motifs in brain networks. *PLoS biology*, 2(11):e369, 2004.

[40] I. Stanton and G. Kliot. Streaming graph partitioning for large distributed graphs. In *KDD*, pages 1222–1230, 2012.

[41] S. Sun, Y. Che, L. Wang, and Q. Luo. Efficient parallel subgraph enumeration on a single machine. In *ICDE*, pages 232–243, 2019.

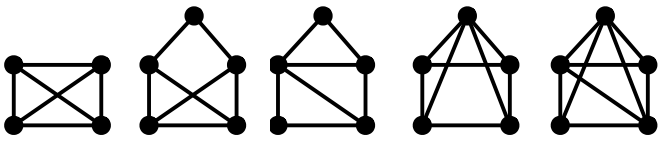
[42] C. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. In *WSDM*, pages 333–342, 2014.

[43] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher. Scalable motif-aware graph clustering. In *WWW*, pages 1451–1460, 2017.

[44] D. Wagner and F. Wagner. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750, 1993.

[45] P. Wang, J. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. Efficiently estimating motif statistics of large networks. *TKDD*, 9(2):8:1–8:27, 2014.

[46] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. In *KDD*, pages 555–564, 2017.



(a) Q_6 (b) Q_7 (c) Q_8 (d) Q_9 (e) Q_{10}

Figure 11: Query motifs used in the experiments.

Dataset	Statistics	Query Motif				
		Q_6	Q_7	Q_8	Q_9	Q_{10}
Hyves	Time (s)	5.0E+1	3.5E+2	2.5E+3	2.5E+2	3.0E+2
	Count	1.8E+6	1.7E+8	9.4E+8	4.9E+7	1.3E+8
Flixster	Time (s)	3.5E+2	1.3E+4	7.6E+4	5.0E+3	4.6E+4
	Count	4.7E+7	1.2E+9	9.5E+9	5.2E+8	3.2E+9
Gowalla	Time (s)	1.5E+2	8.9E+3	7.1E+4	5.0E+3	1.3E+4
	Count	1.2E+7	7.8E+8	7.8E+9	2.3E+8	8.1E+8

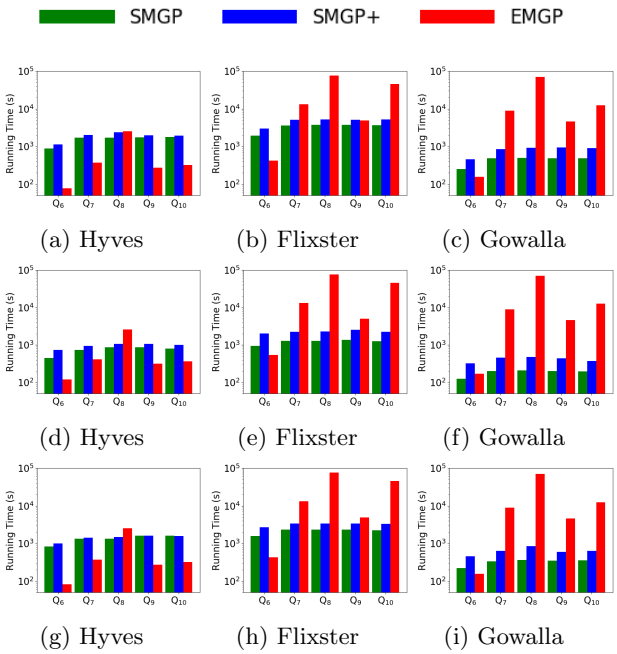
Table 5: Statistics of EMGP.

9. APPENDIX

In this section, we evaluate SMGP, SMGP+ and EMGP on Hyves, Flixster and Gowalla with five more motifs in Figure 11. The statistics of these motifs and corresponding computation cost of EMGP are reported in Table 5. Furthermore, we also conduct experiments on a much larger and denser dataset Livejournal with motif Q_1 and Q_2 in Figure 4.

Performance on Hyves, Flixster and Gowalla. Figure 15, Figure 16 and Figure 17 compares their performance with `Metis`, `MAPPR` and `Fennel` as the underlying GP kernel respectively. SMGP+ notably outperforms SMGP while achieving near-identical partitioning quality compared to EMGP in many cases, which is consistent to our observation in Section 5.1. Figure 12 compares their efficiency. EMGP runs faster than SMGP and SMGP+ on sparse datasets with simple motifs. As the density or the scale of the graph increases, EMGP loses its advantage drastically and can be significantly outperformed by SMGP and SMGP+, which is consistent to our observation in Section 5.2.

Performance on LiveJournal. Figure 13 compares their performance with `MAPPR` and `Fennel` as the underlying GP kernel respectively. We can have observations consistent to the ones in Section 5.1. Note that we are not able to report the results with `Metis` as the GP kernel as the sum of weights of total edges exceeds the maximum value `Metis` allows during the coarsening stage. Figure 14 compares their efficiency. By considering Figure 8 and Figure 14 together, we can observe that EMGP gradually loses the advantage w.r.t. efficiency as the scale and density of the graph grows.

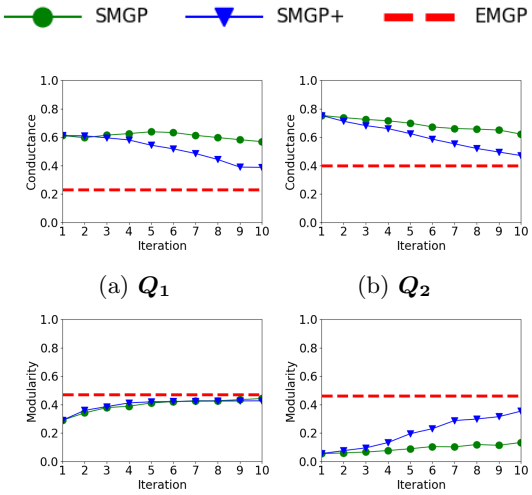


(a) Hyves (b) Flixster (c) Gowalla

(d) Hyves (e) Flixster (f) Gowalla

(g) Hyves (h) Flixster (i) Gowalla

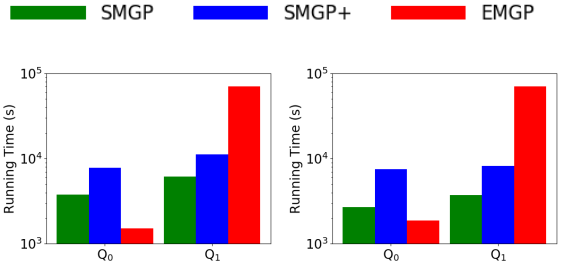
Figure 12: Efficiency comparison with `Metis` (1*st* row), `MAPPR` (2*nd* row) and `Fennel` (3*rd* row) as GP kernels.



(a) Q_1 (b) Q_2

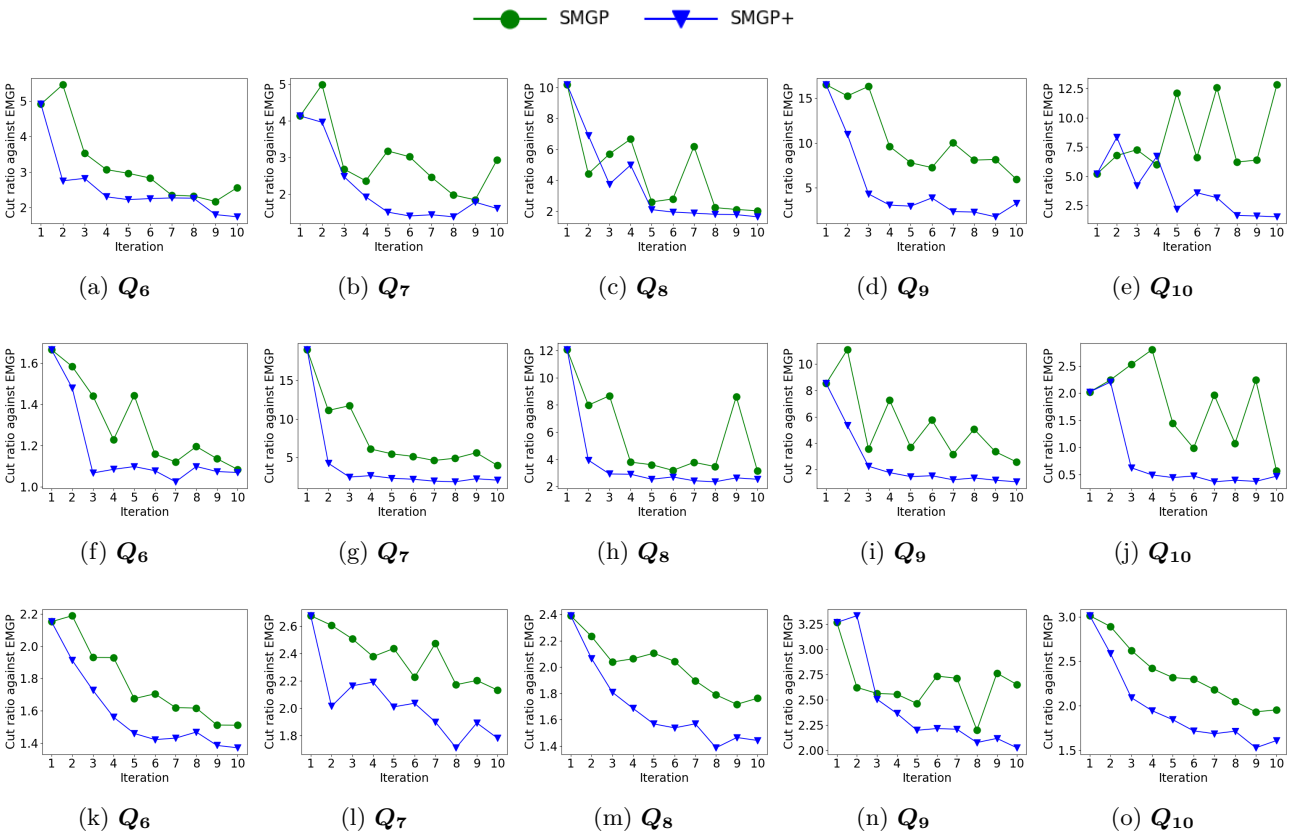
(c) Q_1 (d) Q_2

Figure 13: Performance on LiveJournal with `MAPPR` (1*st* row) and `Fennel`(2*nd* row) as GP kernels.



(a) `MAPPR` (b) `Fennel`

Figure 14: Efficiency comparison on Livejournal with `MAPPR` and `Fennel` as GP kernel.

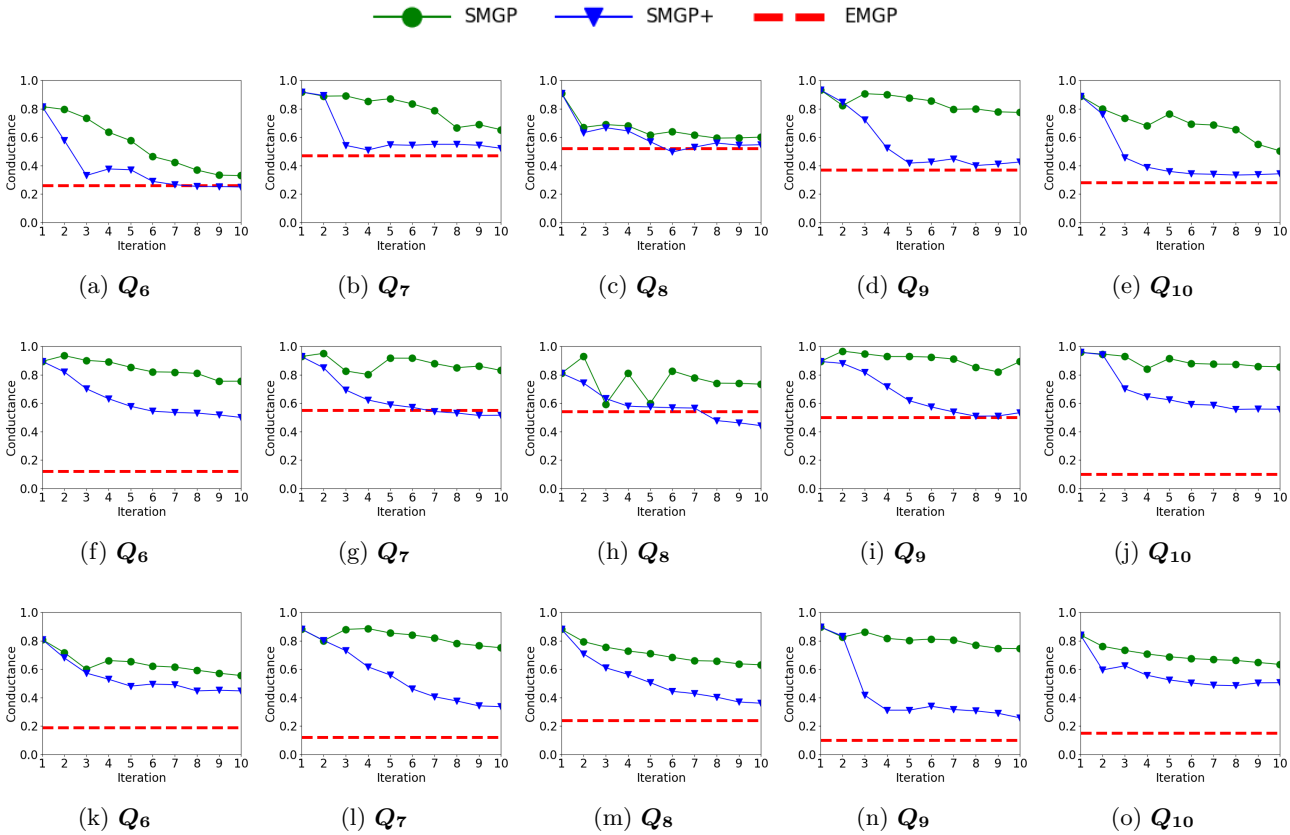


(a) $\boldsymbol{Q_6}$ (b) $\boldsymbol{Q_7}$ (c) $\boldsymbol{Q_8}$ (d) $\boldsymbol{Q_9}$ (e) $\boldsymbol{Q_{10}}$

(f) $\boldsymbol{Q_6}$ (g) $\boldsymbol{Q_7}$ (h) $\boldsymbol{Q_8}$ (i) $\boldsymbol{Q_9}$ (j) $\boldsymbol{Q_{10}}$

(k) $\boldsymbol{Q_6}$ (l) $\boldsymbol{Q_7}$ (m) $\boldsymbol{Q_8}$ (n) $\boldsymbol{Q_9}$ (o) $\boldsymbol{Q_{10}}$

Figure 15: Performance on Hyves (1*st* row), Flixster (2*nd* row) and Gowalla (3*rd* row) with `Metis` as the GP kernel.

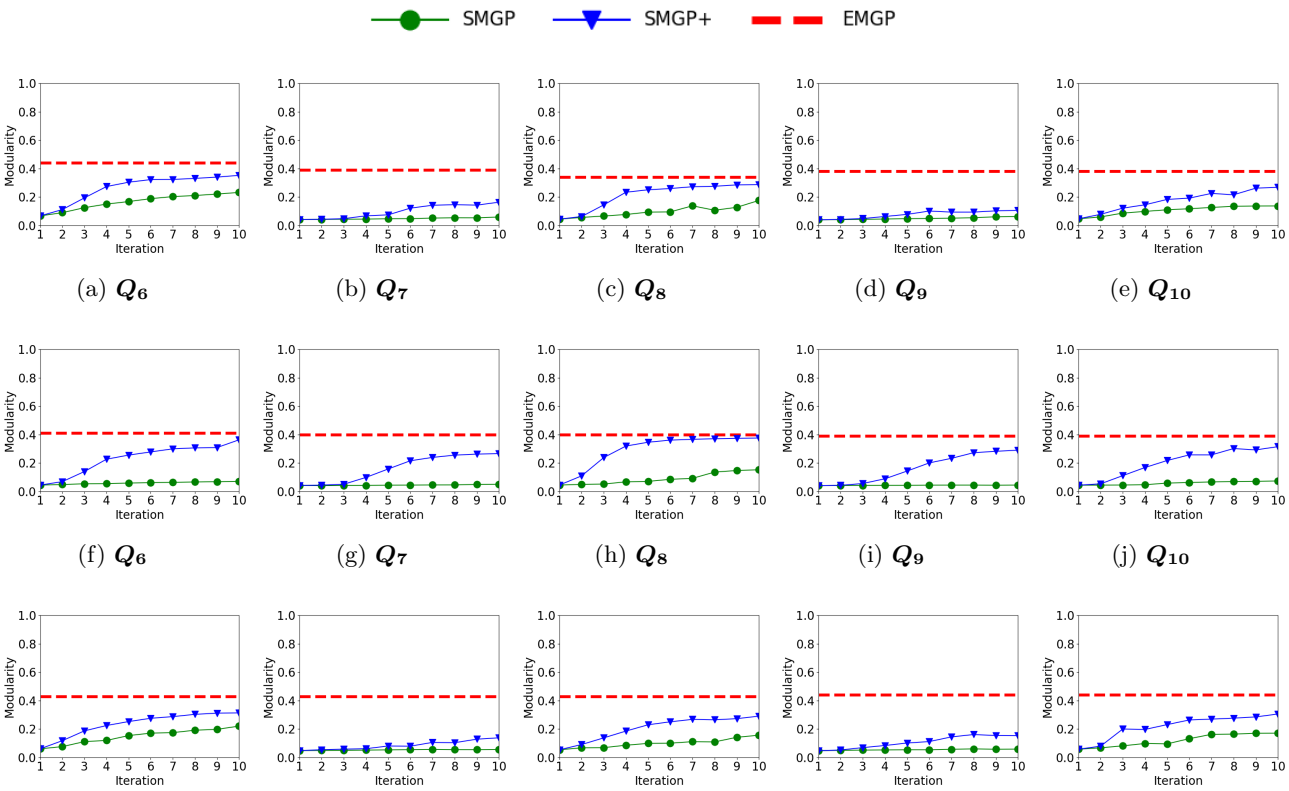


(a) $\boldsymbol{Q_6}$ (b) $\boldsymbol{Q_7}$ (c) $\boldsymbol{Q_8}$ (d) $\boldsymbol{Q_9}$ (e) $\boldsymbol{Q_{10}}$

(f) $\boldsymbol{Q_6}$ (g) $\boldsymbol{Q_7}$ (h) $\boldsymbol{Q_8}$ (i) $\boldsymbol{Q_9}$ (j) $\boldsymbol{Q_{10}}$

(k) $\boldsymbol{Q_6}$ (l) $\boldsymbol{Q_7}$ (m) $\boldsymbol{Q_8}$ (n) $\boldsymbol{Q_9}$ (o) $\boldsymbol{Q_{10}}$

Figure 16: Performance on Hyves (1*st* row), Flixster (2*nd* row) and Gowalla (3*rd* row) with `MAPPR` as the GP kernel.



(a) $\boldsymbol{Q_6}$ (b) $\boldsymbol{Q_7}$ (c) $\boldsymbol{Q_8}$ (d) $\boldsymbol{Q_9}$ (e) $\boldsymbol{Q_{10}}$

(f) $\boldsymbol{Q_6}$ (g) $\boldsymbol{Q_7}$ (h) $\boldsymbol{Q_8}$ (i) $\boldsymbol{Q_9}$ (j) $\boldsymbol{Q_{10}}$

(k) $\boldsymbol{Q_6}$ (l) $\boldsymbol{Q_7}$ (m) $\boldsymbol{Q_8}$ (n) $\boldsymbol{Q_9}$ (o) $\boldsymbol{Q_{10}}$

Figure 17: Performance on Hyves (1*st* row), Flixster (2*nd* row) and Gowalla (3*rd* row) with `Fennel` as the GP kernel.