Balanced Partitioning for Optimizing Big Graph Computation: Complexities and Approximation Algorithms

Baoling Ning[1] and Jianzhong Li[2,3]

[1] Heilongjiang University, China. `ningbaoling2009@163.com`
[2] Harbin Institute of Technology, China.
[3] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China.

Abstract. Graph partitioning is a key fundamental problem in the area of big graph computation. Previous works do not consider the practical requirements when optimizing the big data analysis in real applications. In this paper, motivated by optimizing the big data computing applications, two typical problems of graph partitioning are studied. The first problem is to optimize the performance of specific workloads by graph partitioning, which lacks of algorithms with performance guarantees. The second problem is to optimize the computation of motifs by graph partitioning, which has not been focused by previous works. First, the formal definitions of the above two problems are introduced, and the semidefinite programming representations are also designed based on the analysis of the properties of the two problems. For the motif based partitioning problem, it is proved to be NP-complete even for the special case of $k = 2$ and the motif is a triangle, and its inapproximability is also shown by proving that there are no efficient algorithms with finite approximation ratio. Finally, using the semidifinite programming and sophisticated rounding techniques, the bi-criteria $O(\sqrt{\log n \log k})$-approximation algorithms with polynomial time cost are designed and analyzed for them.

Keywords: Graph Partitioning · Bi-criteria Approximation Algorithm · Semidefinite Programming

1 Introduction

Graph partitioning is a widely used strategy when processing big graph data. On one side, graph partitioning provides fundamental functions for supporting many important graph analysis operations (*e.g.*, *traversal*, *pagerank*, *clustering*). On the other side, graph partitioning is a key fundamental problem in big graph computing platforms, which is the core of many mechanisms such as load balancing. Therefore, the problem of balanced graph partitioning is important in the area of big graph analysis.

In previous works, the most common problem focused by the them is called k-balanced graph partitioning, kBGP for short. Given a graph G and an integer $k \geq 2$, the kBGP problem is to find a solution for partitioning the graph nodes, such that all nodes are partitioned into k disjoint partitions of the same size, and the partitioning cost is minimal. Here, the partitioning cost should be chosen according to the practical requirements, the most popular cost is the size (or weighted size) of the *cut* edges between different partitions. The problem has been proved to be NP-complete, as one of the most classical NP-complete problems found in the 1970s, more precisely, even considering the special case of $k = 2$, the kBGP problem is still NP-complete [12]. Then, a natural and important research issue is try to design an efficient approximation solution, however, in 2006, it is proved by [2] that for arbitrary $k \geq 3$ there are no polynomial-time algorithms for kBGP with finite approximation ratio unless P=NP. Thus, it is almost impossible to design efficient approximation algorithms for kBGP, and the follow-up research works have no way but to relax the performance requirements further. One popular and principal way is to design *bi-criteria approximation algorithms*, where the kBGPproblem is relaxed to the (k, τ)-balanced graph partition problem ($k\tau$BGP for short). Given a graph G, an integer $k \geq 2$ and a real number $\tau \geq 1$, the $k\tau$BGP problem is to find a solution for partitioning the nodes into k partitions of at most $\tau \lceil \frac{|V_G|}{k} \rceil$ size each, such that the partitioning cost is minimal. The meaning of *bi-criteria* is that the output is relaxed to be a solution of the $k\tau$BGP problem but the cost is compared with the optimal solution of the original kBGP problem (equivalently, $\tau = 1$). Current works on *bi-criteria* approximation algorithms can be divided into three kinds. (1) The first kind is

Table 1. The List of Notations

notation	comment
G	the input graph
V_G, E_G	the node and edge set of G
k	the number of partitions
PAR_G	a partition solution of G
PAR_G^k	a k-partition solution of G
V_i	some partition
τ	balance factor
Φ	the given workload
M	the given motif
$\mathsf{cost}_\Phi(\cdot)$	the cost function for optimizing workloads evaluation
$\mathsf{cost}_M(\cdot)$	the cost function for optimizing motif computation

designed based on the algorithms for the 2BGP problem, and the best approximation ratio is $O(\log k\sqrt{\log n})$ [4,18]. (2) The second kind is designed using the mathematical programming techniques, and the best ratio is $O(\sqrt{\log n \log k})$ [14,7]. (3) The third kind is designed based on combinatorial techniques such as dynamic programming, and the best approximation ratio can be $O(\log n)$ [10,2,17].

As shown above, previous works have focused on the graph partitioning problem and produced a plenty of insightful results. However, in view of the applications of big graph processing systems such as [15] and [13], many novel requirements for the partitioning problem have emerged, and current methods can not satisfy the new requirements and does not support the optimization for novel processing of graph data.

Therefore, in this paper, two typical problems of graph partitioning motivated by the big data computing applications are studied. The first problem is to optimize the performance of specific workloads by adjusting the partitioning of graph data. Current works focusing on this problem aim to design heuristic [16] or learning based partitioning methods [9], and one of their drawbacks is the lack of performance guarantees. Our goal is to provide efficient partitioning methods with performance guarantee to optimize the computation of the given workloads. The second problem is to optimize the computation of motifs by adjusting the partitioning of graph data. As far as we know, there still lacks of research works focusing on this problem. Our goal is to provide the fundamental theoretical analysis results and try to design efficient partitioning methods for it. The main contributions of the paper can be summarized as follows.

① The formal definition of the balanced graph partition problem driven by optimizing workloads is introduced. By transforming the problem to a representation of semidefinite program, a bi-criteria $O(\sqrt{\log n \log k})$-approximation algorithm with polynomial time cost is proposed.

② The formal definition of the balanced graph partition problem based on optimizing motif computation is introduced. The problem is proved to be NP-complete even for the special case of $k = 2$ and the motif is a triangle. Then, its inapproximability is also shown by proving that there are no efficient algorithms with finite approximation ratio.

③ For the special case that the motif is a triangle, using the semidefinite programming techniques, a bi-criteria $O(\sqrt{\log n \log k})$-approximation algorithm with polynomial time cost is designed. For the general case, the proposed algorithm is proved to be extended with the same performance guarantee.

2 Preliminaries

In this part, we will first introduce some basic definitions and useful notations, then, by comparing with the classical balanced graph partitioning problem, the problems of balanced graph partitioning driven by optimizing workload and motif are introduced. Some important symbols utilized in the paper are summarized in Table 1.

2.1 Graph and Graph Partition

In this paper, the input graph is denoted by $G = (V, E)$, where V is the node set and $E \subseteq V \times V$ is the edge set. When there are several graphs to be distinguished, we also use V_G and E_G to emphasize that the graph G is referred.

Definition 1 (the partition and k-partition solution of a graph). *Given a graph G, a partition solution is indeed defined on the node set V, that is $\mathrm{PAR}_G = \{V_1, \cdots, V_m\}$. Here, the union of all V_is equals to V, and all V_is are disjoint from each other. If the size of the partition solution is k, that is $m = k$, it is also called to be a k-partition solution, denoted by $\mathrm{PAR}_G^k = \{V_1, \cdots, V_k\}$.* □

Moreover, for a node $u \in V$, we use $\mathrm{PAR}_G(u)$ to represent the partition where the node u is placed. Obviously, if $\mathrm{PAR}_G(u) = V_i$, we always have $u \in V_i$.

With a clean context, we also use PAR_G to represent PAR_G^k. Specially, when k is some constant, for example $k = 2$, 2-partition is utilized for simplicity.

2.2 The Classical Balanced Graph Partitioning Problem

The classical balanced graph partitioning problem is indeed designed for load balancing, whose formal definition is as follows.

Definition 2 (k-balanced partition solution of a graph). *A k-balanced partitioning solution of G is a k-partition solution $\mathrm{PAR}_G = \{V_i\}$ such that for any $i \in [1, k]$ we have $|V_i| \leq \lceil \frac{|V|}{k} \rceil$.* □

Since there are many possible partitioning solution, to select the best one, the optimal partitioning solution depends on the partitioning cost function. The most commonly used is defined based on the cut set, which can be expressed in Equation (1).

$$\mathsf{cost}(\mathrm{PAR}_G) = |\{(u, v) \in E \mid \mathrm{PAR}_G(u) \neq \mathrm{PAR}_G(v)\}| \tag{1}$$

Then, to minimize the size of cutting edges, we have the following definition.

Definition 3 (k-balanced graph partitioning, kBGP for short). *Given a graph G and an positive integer k, the k-balanced graph partitioning problem is to find a k-balanced partition solution PAR_G for G such that the cost $\mathsf{cost}(\mathrm{PAR}_G)$ is minimized.* □

The kBGP problem is one of the most classical NP-complete problems[11], which is also called the graph bisection problem for the special case $k = 2$.

Theorem 1 ([11]). *The $kBGP$ problem is NP-complete, even for the special case $k = 2$.* □

Even worse, we have the following result.

Theorem 2 ([2]). *For any $k \geq 3$, unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial time algorithm for the $kBGP$ problem with finite approximation ratio.* □

Due to the hardness shown by Theorem 2, more relaxations are considered and the *bi-criteria* approximation algorithms are designed.

Definition 4 (the (k, τ)-balanced partition solution of a graph). *Given a balance factor τ, a (k, τ)-balanced partition solution of a graph G is a k-partiton solution $\mathrm{PAR}_G = \{V_i\}$, such that for any $i \in [1, k]$ we have $|V_i| \leq \tau \lceil \frac{|V|}{k} \rceil$.* □

Given a kBGP instance (G, k), the aim of a bi-criteria (k, τ) approximation algorithm $\mathcal{P}$ is to find a (k, τ)-balanced partitioning solution PAR for G such that the approximation ratio of $\mathcal{P}$ is defined as $\rho_{\mathcal{P}} = \frac{\mathsf{cost}(\mathrm{PAR})}{\mathsf{cost}(\mathrm{PAR}^*)}$ where PAR^* is the optimal solution for the kBGP problem. It should be noted that, because of the definition of bi-criteria approximation algorithms, the performance ratio $\rho_{\mathcal{P}}$ may be less than 1.

2.3 Workload Driven Balanced Graph Partitioning Problem

In big data computing platforms, there are many different kinds of computing tasks, which may require different data and have different costs of computation, communication and so on. Therefore, a key problem is to determine the partitioning solution to optimize specific workloads. In this part, considering several kinds of typical computing tasks such as pattern matching, traversal, shortest path and so on, the problem of balanced graph partitioning driven by workloads is introduced.
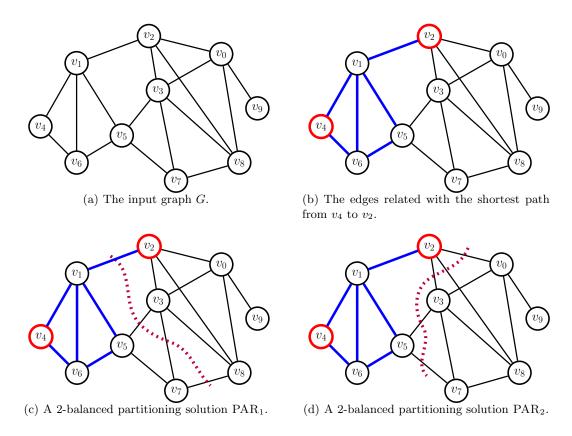

(a) The input graph G.

(b) The edges related with the shortest path from v_4 to v_2.

(c) A 2-balanced partitioning solution PAR_1.

(d) A 2-balanced partitioning solution PAR_2.

Fig. 1. An Example of Balanced Graph Partitioning Driven by Workload

Example 1. Consider the graph G shown in Fig. 1(a), which is composed of 10 nodes. Let $k = 2$, that is we need to partition the nodes in G into two parts. Two possible partitioning solution are shown in Fig. 1(c) and 1(d), denoted by PAR_1 and PAR_2. Here, the solution of PAR_1 is $\{v_1, v_4, v_5, v_6, v_7\}$ and $\{v_0, v_2, v_3, v_8, v_9\}$, whose cut edges are $\{(v_1, v_2), (v_3, v_5), (v_3, v_7), (v_7, v_8)\}$ and the cost is 4, that is $\mathsf{cost}(PAR_1) = 4$. For the solution PAR_2, the partitioning result is $\{v_1, v_2, v_4, v_5, v_6\}$ and $\{v_0, v_3, v_7, v_8, v_9\}$, whose cut edges eare $\{(v_0, v_2), (v_2, v_8), (v_2, v_3), (v_3, v_5), (v_5, v_7)\}$ and the cost is 5, that is $\mathsf{cost}(PAR_1) = 5$. According to Definition 3, PAR_1 is better than PAR_2. However, if we consider some possible specific computing jobs, the result may be that PAR_1 is worse than PAR_2. Assume that the shorted path between v_4 and v_2 needs to be computed on G, and it is assumed that the algorithm needs to do a breath first travel beginning from v_4. Then, it is easy to verify that the visited edges are the ones with blue color in Fig. 1(b). Comparing PAR_1 with PAR_2 again, it can be found that PAR_2 is better than PAR_1. As shown in Fig. 1(c), PAR_1 places the nodes of the edge (v_1, v_2) into different partitions, which may cause once communication cost, while as shown in Fig. 1(d), PAR_2 does not cut any edges which will be visited by the algorithm for shortest path, the communication cost will be 0. □

As shown by Example 1, the cost function utilized in Definition 3 is not proper for optimizing the specific workloads. Next, based on a formal representation of workload, the problem of workload driven k-balanced graph partitioning, WkBGP for short, is introduced.

The workload on graph G can be represented by a set $\Phi = \{(\phi_1, c_1), \cdots, (\phi_w, c_w)\}$, where each ϕ_i is a specific computing job in the workload, c_i is a positive integer indicating the frequency that ϕ_i appears. One job ϕ_i is a set $\{(e_{ij}, c_{ij})\}$, where $e_{ij} \in E_G$ is an edge of G and c_{ij} is a positive integer. The element $(e_{ij}, c_{ij}) \in \phi_i$ means that the job ϕ_i needs to visit the edge e_{ij} for c_{ij} times, and if the two nodes are placed on two different partitions, there may be c_{ij} communications.

Given a partitioning solution PAR and a workload representation Φ on G, let $\mathsf{cost}_\Phi(\mathrm{PAR})$ be the expected communication cost caused by partitioning G according to PAR and computing Φ on G, which can be expressed by the following form.

$$\mathsf{cost}_\Phi(\mathrm{PAR}) = \frac{1}{\sum_{1 \le i \le w} c_i} \sum_{1 \le i \le w} \big(c_i \cdot \mathsf{cost}_{\phi_i}(\mathrm{PAR})\big) \tag{2}$$
$$= \frac{1}{\sum_{1 \le i \le w} c_i} \sum_{1 \le i \le w} \left(c_i \cdot \sum_{1 \le j \le |\phi_i|} \big(c_{ij} \cdot \mathbf{I}(\mathrm{PAR}(u_{ij}) \ne \mathrm{PAR}(v_{ij}))\big)\right)$$

Here, $\mathsf{cost}_{\phi_i}(\mathrm{PAR})$ is the communication cost needed to evaluate the job ϕ_i, the two nodes related with the edge e_{ij} are u_{ij} and v_{ij}, and $\mathbf{I}(\cdot)$ is a function whose value is 1 if the required condition is satisfied, otherwise it is 0.

Definition 5 (The WkBGP Problem). *Given an input graph G, a positive integer k and a workload set Φ, the WkBGP problem is to find a k-balanced partitioning solution PAR_G for G such that the cost $\mathsf{cost}_\Phi(\mathrm{PAR}_G)$ of computing Φ is minimized.* □

It is easy to verify that the kBGP is a special case of the WkBGP problem, according to Theorem 1, we have the following result trivially.

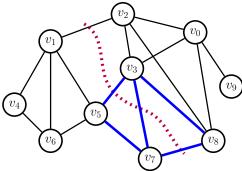**Proposition 1.** *The WkBGP problem is NP-complete.* □

Then, the inapproximability result of kBGP can be extended to the WkBGP problem naturally.

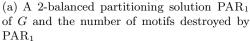**Proposition 2.** *Unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial time algorithm for WkBGP with finite approximation ratio.* □
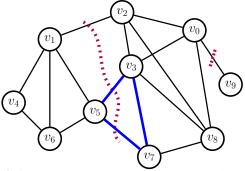
2.4   Motif Based Balanced Graph Partitioning Problem

Motif computation is a hot topic in the area of graph data processing, which has become a typical kind of computing tasks in big data platforms. The motivation of studying motif computation comes from the traditional graph matching problem, which is to determine whether a given pattern graph Q appears in a given data graph G. Since the graph matching problem is intractable, the most popular and feasible solution is to design efficient algorithms for the pattern graphs with small sizes. Motif is just the graph of small sizes, such as 3 or 4 nodes. In this part, the motif based balanced graph partitioning problem (MkBGP for short) is introduced.

Example 2. Consider the graph G shown in Fig. 1(a), and let $k = 2$, that is we need to compute a 2-balanced partitioning solution. There are two possible 2-balanced partitioning solutions shown in Fig. 2(a) and 2(b). The partitioning solution shown in Fig. 2(a) is the same as the one shown in Fig. 1(c), we still denote it by PAR_1. The solution shown in Fig. 2(b) is denoted by PAR_3. In PAR_1, the graph G is partitioned into two sets $\{v_1, v_4, v_5, v_6, v_7\}$ and $\{v_0, v_2, v_3, v_8, v_9\}$, the corresponding cut edges are $\{(v_1, v_2), (v_3, v_5), (v_3, v_7), (v_7, v_8)\}$, and the partitioning cost is 4, that is $\mathsf{cost}(\mathrm{PAR}_1) = 4$. In PAR_3, the graph G is partitioned into two sets $\{v_1, v_4, v_5, v_6, v_9\}$ and $\{v_0, v_2, v_3, v_7, v_8\}$, the corresponding cut edges are $\{(v_1, v_2), (v_3, v_5), (v_5, v_7), (v_0, v_9)\}$, and the partitioning cost is also 4, that is $\mathsf{cost}(\mathrm{PAR}_3) = 4$. It can be found that, using the traditional

(a) A 2-balanced partitioning solution PAR_1 of G and the number of motifs destroyed by PAR_1

(b) A 2-balanced partitioning solution PAR_3 of G and the number of motifs destroyed by PAR_3

Fig. 2. An Example of Balanced Graph Partitioning Driven by Motif Computation

partitioning cost measure, no one of the two solutions is better than the other one. Next, let us consider a specific job of motif computation, whose aim is to calculate the number of triangles in G. Obviously, no matter which method is utilized, if the three nodes of one triangle are not in one partition, there will be at least one communication. In Fig. 2(a) and 2(b), the triangles destroyed by the partitioning solution are colored with blue. Then, by counting the number of triangles destroyed, the solution PAR_3 should be better than PAR_1. □

According to Example 2, it can be found that the $\mathsf{cost}(\cdot)$ function used by Definition 3 is based on counting the cut edges and it is not proper for optimizing the tasks of motif computation. Next, the cost function for motif computation is introduced first, and then the definition of the motif computation based k-balanced graph partitioning (MkBGP for short) is introduced.

Let c be a constant, a c-Motif is a graph with c nodes. Given a partitioning solution PAR of G and a motif M, intuitively, let $\mathsf{cost}_M(\mathrm{PAR})$ be the communication cost of processing computation of M on partitions obtained according to PAR.

$$\mathsf{cost}_M(\mathrm{PAR}) = \sum_{H \in G\langle M \rangle} \mathbf{I}\big(\exists (u,v) \in H \mathrm{s.t.PAR}(u) \neq \mathrm{PAR}(v)\big) \tag{3}$$

Here, $G\langle M \rangle$ means all appearance of the motif M in the graph G, and $\mathbf{I}(\cdot)$ is the indicator function whose value is in $\{0,1\}$.

Definition 6 (The MkBGP problem). *Given a graph G, a positive interger k and a c-Motif M, the MkBGP problem is to compute a k-balanced partitioning solution PAR_G for G such that the cost $\mathsf{cost}_M(\mathrm{PAR}_G)$ is minimal.* □

Different from the WkBGP problem, the kBGP problem is not a special case of MkBGP, therefore, the complexity result of MkBGP can not be obtained directly from Theorem 1.

3 Graph Partitioning Algorithm for Workload Optimization

3.1 Semidefinite Programming

Let $\boldsymbol{X}$ be a symmetric square $n \times n$ matrix. If all eigenvalues of $\boldsymbol{X}$ are non positive, it is also called a positive semidefinite matrix, psd for short.

According to the characteristics of semidefinite matrix, the matrix $\boldsymbol{X}$ can be represented by $\boldsymbol{X} = \boldsymbol{V}^T \boldsymbol{V}$ also, where $\boldsymbol{V}$ is a $m \times n$ matrix. If each column of the matrix $\boldsymbol{V}$ is treated as a m-dimensional vector,

that is $\boldsymbol{V}$ is composed of n vectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, ..., $\boldsymbol{v}_n$. Then, each element x_{ij} in $\boldsymbol{X}$ can be represented by $\sum_{1 \le h \le m} v_{ih} v_{hj}$. Obviously, x_{ij} can be represented by $\boldsymbol{v}_i^T \boldsymbol{v}_j$ or $\boldsymbol{v}_i \cdot \boldsymbol{v}_j$. Therefore, a typical semidefinite program can be represented by the following form.

$$
\text{minimize} \quad \sum_{i,j} c_{ij}(\boldsymbol{v}_i \cdot \boldsymbol{v}_j) \tag{4}
$$

$$
\text{s.t.} \quad \sum_{i,j} a_{ij}^k (\boldsymbol{v}_i \cdot \boldsymbol{v}_j) = b_k, \qquad \forall k \in [1, l]
$$

$$
\boldsymbol{v}_i \in \mathbb{R}^m, \qquad \forall i \in [1, n]
$$

According to [1,6], given a SDP of size n and $\epsilon > 0$, there exists a $O\big((n+1/\epsilon)^{O(1)}\big)$-time algorithm solving the SDP with an approximation ratio $(1 + \epsilon)$.

3.2 The Semidefinite Program for WkBGP

In this part, we will represent the WkBGP problem using semidefinite program. The intuitive idea can be explained as follows. Each node in V can be represented as a m-dimensional vector, where $m \ge k$, and it is denoted by $\boldsymbol{v}$. Ideally, it is expected that the vector $\boldsymbol{v}$ can indicate which partition the node should be placed into. When two nodes are in the same partition, the corresponding vectors should be same or very similar. Given the input (G, k, Φ) of the WkBGP problem, the semidefinite representation can be expressed as Equation (5).

$$
\min \quad \frac{1}{\sum_{1 \le i \le w} c_i} \sum_{1 \le i \le w} \left(c_i \cdot \sum_{1 \le j \le |\phi_i|} \left(c_{ij} \cdot \frac{1}{2} \|\boldsymbol{u}_{ij} - \boldsymbol{v}_{ij}\|_2^2 \right) \right) \tag{5}
$$

$$
\text{s.t.} \quad \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \|\boldsymbol{v} - \boldsymbol{w}\|_2^2 \ge \|\boldsymbol{u} - \boldsymbol{w}\|_2^2 \qquad \forall u, v, w \in V \tag{6}
$$

$$
\sum_{v \in S} \frac{1}{2} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \ge |S| - \frac{n}{k} \qquad \forall S \subseteq V, u \in S \tag{7}
$$

Here, $\| \cdot \|_2^2$ used in Equation (5) is the square of $L2$ normal form, that is ℓ_2^2, which can be defined as $\|\boldsymbol{x}\|_2^2 = \sum_{i=1}^m x_i^2$. The first constraint shown by Equation (6) is the triangle inequality defined on ℓ_2^2, which is firstly proposed by [4] and used to express the partitioning requirements. The second constraint shown by Equation (7) is also called the spreading constraint, which is proposed by [8] and used to control the size of each partition. The optimizing goal shown in Equation (5) is a variant of the cost function $\mathsf{cost}_\Phi(\cdot)$ utilized in the definition of the WkBGP problem, where the part $\mathbf{I}(\text{PAR}(u_{ij}) \ne \text{PAR}(v_{ij}))$ of $\mathsf{cost}_\Phi(\cdot)$ is replaced with $\frac{1}{2}\|\boldsymbol{u}_{ij} - \boldsymbol{v}_{ij}\|_2^2$. Intuitively, for two nodes u_{ij} and v_{ij}, if the distance between $\boldsymbol{u}_{ij}$ and $\boldsymbol{v}_{ij}$ is too large (like ≈ 2), it means that $\text{PAR}(u_{ij}) \ne \text{PAR}(v_{ij})$, otherwise, it means that $\text{PAR}(u_{ij}) = \text{PAR}(v_{ij})$. Therefore, the optimizing goal shown by Equation (5) is indeed a relaxed form of the cost function $\mathsf{cost}_\Phi(\cdot)$, and the transformation is roughly equivalent.

It should be noted that, given an instance of the WkBGP problem, the semidefinite program shown in Equation (5) will not be constructed explicitly, because the size of the constraints is huge. More specifically, the goal of Equation (5) can be built within $O(|\Phi|)$, and the constraints in Equation (6) are built with respect to each triple of nodes in G and it can be finished within $O(|V_G|^3)$ obviously. However, the constraints shown in Equation (7) are built based on each possible subset of V_G, therefore, the size of the constraints can reach $\Theta(|V_G| \cdot 2^{|V_G|})$, which can not be built within polynomial time. A feasible way to process the above constraints is to built a specific constraint when it is needed, since the constraint can be defined over every possible subset of V_G and it can be checked efficiently. Therefore, the semidefinite program is solved based on the optimizing goal and the constraints in Equation (6) first, and then for each $v \in V_G$ the corresponding constraint can be checked as the following way. First, all nodes like $u \in V \setminus \{v\}$ are computed, and then the corresponding distance $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$ is calculated and the nodes will be sorted into an increasing order. Finally, the nodes will be added into the set S and checked whether the condition defined by Equation (7) is satisfied. According to Lemma **??**, we have the following result.

Theorem 3. *The semidefinite program shown in Equation* (5) *can be solved in polynomial time.* □

Proof. Let N be the size of the input instance of the WkBGP problem, where n is the number of nodes in G. Obviously, we have $N = \max\{n, |\varPhi|\}$. Let us consider the range value of the optimized goal of Equation (5). First, the part related with c_i can be treated as the weighted sum of the expression $\sum_{1 \le j \le |\phi_i|}(c_{ij} \cdot \frac{1}{2}\|\boldsymbol{u}_{ij} - \boldsymbol{v}_{ij}\|_2^2)$, and it can not be larger than $\max\{\sum_{1 \le j \le |\phi_i|}(c_{ij} \cdot \frac{1}{2}\|\boldsymbol{u}_{ij} - \boldsymbol{v}_{ij}\|_2^2) \mid i \in [1, w]\}$. Let γ be the constant $\max\{\sum_{1 \le j \le |\phi_i|} c_{ij} \mid i \in [1, w]\}$. If the optimizing goal of Equation (5) is divided by γ, which will not change the optimized partitioning solution obviously, the optimizing goal after the division will not be larger than the weighted sum of the distance $\frac{1}{2}\|\boldsymbol{u}_{ij} - \boldsymbol{v}_{ij}\|_2^2$ over all edges $(u_{ij}, v_{ij}) \in E_G$. Then, let the edge with the maximized distance be (u_{max}, v_{max}) and the corresponding distance be $\frac{1}{2}\|\boldsymbol{u}_{max} - \boldsymbol{v}_{max}\|_2^2$. As a result, the final goal can be limited within the range $[0, |E_G| \cdot \frac{1}{2}\|\boldsymbol{u}_{max} - \boldsymbol{v}_{max}\|_2^2]$.

Moreover, let us consider a trivial solution for the semidefinite program. First, randomly select $\frac{n}{k}$ nodes of V and represent them as $v_1, \cdots, v_{n/k}$. The corresponding k-dimensional vectors is set to be $\boldsymbol{v}_i = [\sqrt{n}, 0, 0, \cdots, 0]^T$. Then, repeat the above operations until all nodes in V have been selected, for each step, the value $\sqrt{n}$ is moved backward one position. Since the solution should be m-dimensional vectors, for the other $m - k$ dimensions, let the corresponding value be 0 always.

It is not hard to verify that the above trivial solution satisfy the two groups of constraints. First, for two given nodes u and v, the corresponding vector distance $\frac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$ only has two possible values, 0 or n, and the value is 0 if and only if the two nodes are selected in the same step during the above construction. Then, for the constraints shown in Equation (6), it can be verified by analyzing the distances among three nodes, which have the following two cases. (1) The three nodes have same vectors, then the values of $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$, $\|\boldsymbol{u} - \boldsymbol{w}\|_2^2$ and $\|\boldsymbol{w} - \boldsymbol{v}\|_2^2$ are all 0, the constraints are satisfied trivially. (2) There are at least two nodes having different vectors. Without loss of generality, we assume that $\boldsymbol{u}$ and $\boldsymbol{v}$ are different, then, at least two of the three values $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$, $\|\boldsymbol{u} - \boldsymbol{w}\|_2^2$ and $\|\boldsymbol{w} - \boldsymbol{v}\|_2^2$ are $2n$. Then, the constraints can be satisfied too. Second, to verify the constraints shown in Equation (7) are satisfied, let us consider two cases based on the size of S. (1) The first case is $|S| \le \frac{n}{k}$. For this case, the right side of the constraint satisfies $|S| - \frac{n}{k} \le 0$ and the left side is at least 0. Therefore, the constraints can be satisfied. (2) The second case is $|S| > \frac{n}{k}$. For this case, since during the construction of the trivial solution each step only selects at most n/k nodes, in S, there must be at least one node v whose vector is different from the one of u. Then, for the left side of the constraint, we have $\sum_{v \in S} \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \ge n$, and for the right side, there must be $|S| - \frac{n}{k} \le n - \frac{n}{k} < n$. Thus, the condition $\sum_{v \in S} \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \ge |S| - \frac{n}{k}$ can be satisfied obviously.

Because the trivial solution constructed is a feasible solution, the corresponding cost is an upper bound of the optimize cost, then the value of the optimizing goal can be set within the range $[0, |V_G| \cdot |E_G|]$. Further, according to Lemma **??**, by setting the parameter ϵ to satisfy $\epsilon < \frac{1}{|V_G| \cdot |E_G|}$, the $(1 + \epsilon)$-approximation solution of the corresponding semidefinite program is an exact solution in fact. Finally, the time cost of can be bounded by a polynomial of $(n + |V_G| \cdot |E_G|)$. The proof is finished. □

3.3 SDP Based Approximation Algorithm for WkBGP

This part introduces the partitioning algorithm solving the WkBGP problem, which computes an approximation partitioning solution for the given WkBGP instance using the semidefinite programming and a well designed rounding techniques. The performance guarantee of the approximation solution can be bounded based on analyzing the effects of the rounding method.

Given the workload input $\varPhi$, different jobs may use same edges, obviously, in a partitioning solution, it should be determined whether two nodes are placed in the same partition. To use the semidefinite programming to solve the WkBGP problem, by combining the variables related with the same edge, we can rewritten the problem by the following Equation (8), which can be verified to be equivalent to Equation (5).

Algorithm 1: WKBGPARTITION (Workload driven k Balanced Graph Partitioning)

Input: A Graph $G = \{V, E\}$, an integer k and a workload set Φ.
Output: A partition $\text{PAR}_G = \{V_1, V_2, \cdots, V_k\}$ for G.

1 function WKBGPARTITION(G, k, Φ)
2   Initialize a $|V_G| \times |V_G|$ array W to be 0;
3   Let $C = \sum_{1 \leq i \leq w} c_i$;
4   **for** *each* $(\phi_x, c_x) \in \Phi$ **do**
5   **for** *each* $(e_{xy}, c_{xy}) \in \phi_x$ **do**
6   Let the two nodes of e_{xy} be v_i and v_j;
7 $W[i][j] += \frac{c_x \cdot c_{xy}}{C}$;
8 $W[j][i] = W[i][j]$;

9   Build sdp (8) using $\sum_{(v_i, v_j) \in E_G} W[i][j] \cdot \frac{1}{2} \|v_i - v_j\|_2^2$ as the goal;
10   Let $opt = \{v_1, \cdots, v_n\}$ be the optimal solution of sdp (8);
11   Invoke k-PARTITION of [14] on opt to compute $\text{PAR}_G = \{V_1, \cdots, V_k\}$;
12   **return** PAR_G;

$$\min \quad \sum_{(v_i, v_j) \in E_G} \left(W_{ij} \cdot \frac{1}{2} \|v_i - v_j\|_2^2 \right) \tag{8}$$

$$\text{s.t.} \quad \|u - v\|_2^2 + \|v - w\|_2^2 \geq \|u - w\|_2^2 \qquad \forall u, v, w \in V$$

$$\sum_{v \in S} \frac{1}{2} \|u - v\|_2^2 \geq |S| - \frac{n}{k} \qquad \forall S \subseteq V, u \in S$$

$$W_{ij} = \sum_{1 \leq x \leq w} \sum_{1 \leq y \leq |\phi_x|} \frac{c_x \cdot c_{xy} \cdot \mathbf{I}(e_{xy} = (v_i, v_j))}{C}$$

$$C = \sum_{1 \leq i \leq w} c_i$$

Next, the WKBGPARTITION algorithm for solving the WkBGP problem is introduced, whose details can be found in Algorithm 1. The main procedure of WKBGPARTITION is to first construct the semidefinite program according to Equation (8) based on the input graph G, the workload Φ and the integer k, then use the program solver to obtain the optimized solution for the relaxed variant, finally, computes the result partitioning solution by rounding the vectors obtained in the optimized solution. The transformation between the two semidefinite programs is done under the help of a $|V_G| \times |V_G|$ array W. Specifically, WKBGPARTITION scans all jobs maintained in the workload set Φ, and adds the corresponding weight related with each edge used in the computing jobs into W (line 2-8). Then, WKBGPARTITION utilizes W to construct the semidefinite program shown in Equation (8) (line 9). Let $opt = \{v_1, \cdots, v_n\}$ be the optimized solution for the obtained program, according to Theorem 3, opt can be computed within polynomial time (line 10). Finally, using the rounding techniques proposed in [14] (line 11), the final partitioning solution can be obtained. In details, [14] proposes a rounding method named k-PARTITION, which can output a partitioning solution based on opt as follows. First, k-PARTITION utilizes the space embedding techniques proposed by [3] to map the optimized solution from ℓ_2^2 to ℓ_2, which will enlarge the solution with a ratio at most $O(\sqrt{\log n})$. Then, using the random hyperplane rounding techniques (see [6]), the partitions with size at most $(1 + 2\epsilon)\frac{n}{k}$ can be built and extracted one by one, until all nodes have been processed once.

Next, we will prove the performance guarantee of the approximation ratio of WKBGPARTITION.

Theorem 4. *Given the input G, k and Φ of the WkBGP problem, the cost of the optimal solution of the Equation* (5) *(or* (8)*) is a lower bound of the partitioning cost of the optimal solution of WkBGP.*   □

Proof. Let PAR_G^* be the optimal partitioning solution of the given instance of WkBGP, and let the corresponding partitioning cost be $\text{cost}_\Phi(\text{PAR}_G^*)$. Then, assume that the optimal solution of the corresponding semidefinite program is $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n\}$ and the corresponding optimal value of the goal is A^*. We can build a feasible solution $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n\}$ of the program based on PAR_G^*, whose details can be explained as follows. Let PAR_G^* be the corresponding k-balanced partitioning solution $\{V_1, V_2, \cdots, V_k\}$, where for each V_i we have $|V_i| = \frac{n}{k}$. Then, for any node $v_i \in V_j$, let v_i be the following vector.

$$\boldsymbol{v}_i = [0, \cdots, 1, \cdots, 0]^T \tag{9}$$
$$\uparrow$$
$$\text{the } j\text{th bit}$$

Next, checking the constraints in Equation (5), we can verify the correctness of the solution obtained.

(1) For the constraints in Equation (6), since the $\|\cdot\|_2^2$ distance between two nodes must be 0 or 2, the only possible way to violate the constraints is that both $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$ and $\|\boldsymbol{v} - \boldsymbol{w}\|_2^2$ are 0 but $\|\boldsymbol{u} - \boldsymbol{w}\|_2^2$ is 2. For this case, the nodes u and v are placed into a same partition, u and v belong to one partition, and u and w are placed into two different partitions. Obviously, it is impossible that the above three statements are true at the same time. Therefore, the constraints in Equation (6) must have been satisfied.

(2) For the constraints in Equation (7), let us consider the set S with a fixed size. Obviously, the right side of the constraint is a fixed value. For the left side, considering a given node $u \in S$, the corresponding value of the left side depends on the value of $\frac{1}{2}\|\boldsymbol{v} - \boldsymbol{u}\|_2^2$. Then, it is expected that $\frac{1}{2}\|\boldsymbol{v} - \boldsymbol{u}\|_2^2$ takes a rather small value. Because $\frac{1}{2}\|\boldsymbol{v} - \boldsymbol{u}\|_2^2 = 0$ means that the nodes u and v belong to the same partition, at most $\frac{n}{k} - 1$ nodes can satisfy this condition. Assume that all nodes satisfying the above condition are placed into S, then both sides of Equation (7) are 0 and the constraint is satisfied. If the size of S increases further, the value of the right side will be increased by 1, at the same time, the value of the left side will be increased at least by 1. Therefore, the constraints can be satisfied still.

Then, let us consider the relationship of the optimal values between the semidefinite program and the WkBGP problem. For the feasible solution $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n\}$ constructed, the corresponding value of the optimizing goal of Equation (5) is represented by A. For any two nodes v_i and v_j, it is not hard to verify that $\mathbf{I}(\text{PAR}_G^*(v_i) \neq \text{PAR}_G^*(v_j)) \equiv \frac{1}{2}\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2$, that is they have the same value indeed. According to Equation (2) and (5), we have $\text{cost}_\Phi(\text{PAR}_G^*) = A$. Because A^* is the optimal solution of the semidefinite program, obviously, we have $\text{cost}_\Phi(\text{PAR}_G^*) \geq A^*$. The proof is finished. $\square$

Theorem 4 indicates that the optimal solution of Equation (5) is a lower bound for the cost of the optimal partitioning solution. Then, we need the upper bound.

Lemma 1 ([14]). *Based on the optimal solution of the semidefinite program, the rounding method k-PARTITION will return a partitioning solution with approximation ratio $O(\sqrt{\log k \log n})$ in polynomial time, and the size of each partition can be bounded by $\frac{2n}{k}$.* $\square$

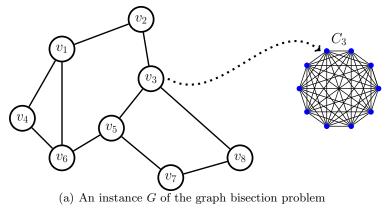Then, we have the following result.

Theorem 5. *The expected time cost of* WKBGPARTITION *shown in Algorithm 1 is polynomial with the input size, and* WKBGPARTITION *is a bi-criteria $(k, 2)$-approximation algorithm with ratio bounded by $O(\sqrt{\log k \log n})$.* $\square$

Proof. According to Lemma 1, Theorem 4 and the details of WKBGPARTITION, it is easy to verify the correctness of the theorem. $\square$

4 Theoretical Analysis of Motif Based Balanced Graph Partitioning Problem

4.1 Analysis of Computational Complexities

In this part, the computational complexities of the MkBGP problem is analyzed and we have the following result.
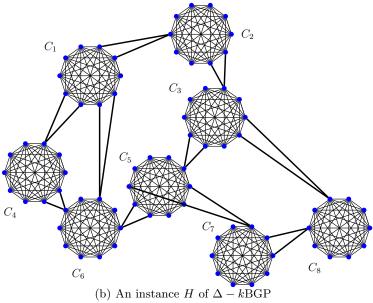
(a) An instance G of the graph bisection problem

(b) An instance H of $\Delta - k$BGP

Fig. 3. An Example of P-time reduction used in the proof of Theorem 7

Theorem 6. *The MkBGP problem is* NP-*complete, even if* $k = 2$. □

Obviously, the proof of Theorem 6 should consider some special motif structures, that is the parameter M in Definition 6 needs to be specified. When the motif M is composed of at most 2 nodes, the MkBGP problem will be trivial or equivalent to kBGP, therefore, we use the motif with 3 nodes to analyze the complexities of MkBGP. Assuming that the motif M is a simple triangle, we can prove Theorem 7, which will imply the lower bound result in Theorem 6, that is the MkBGP problem is NP-hard.

Theorem 7. *Even if* $k = 2$ *and the motif* M *is a triangle, the MkBGP problem is still* NP-*complete.* □

Proof. We use $\Delta-2$BGP to represent the special case of MkBGP when $k = 2$ and the input motif M is a triangle. Then, the proof can be divided into the following two parts. (1) The upper bound: $\Delta-2$BGP is in NP. (2) The lower bound: $\Delta-2$BGP is NP-hard.

The proof will utilize the following decision version of $\Delta-2$BGP. Given a graph G and an integer C, the problem is to determine whether there is a 2-balanced partitioning solution PAR for G such that $\mathsf{cost}_\Delta(\mathrm{PAR}) \leq C$, where $\mathsf{cost}_\Delta(\cdot)$ is the variant of $\mathsf{cost}_M(\cdot)$ when M is a triangle. When the context is clear, we still use $\Delta-2$BGP to represent the decision version.

11

For the upper bound, a NP algorithms for $\Delta-2$BGP will be introduced to show that $\Delta-2$BGP is in NP. The algorithm can be explained as follows. (1) Guess a node set $S \subseteq V$ of size $|V|/2$, and initialize a counter count using 0 which is used to compute the number of triangles destroyed by the partitioning solution. (2) Enumerate all triples (u, v, w) of the node set V, if the corresponding three edges (u, v), (u, w) and (v, w) belong to E and the three nodes are not all in S or $V \setminus S$, the counter count will be increased by 1. (3) After all triples have been processed as above, if the counter satisfies count $\leq C$, the answer 'yes' will be returned, otherwise, the answer 'no' will be returned.

The correctness of the above algorithm can be verified easily, and its time cost can be bounded by $O(|V|^3)$ obviously. Then, it has been proved that the $\Delta-2$BGP problem belongs to NP.

For the lower bound, we given a polynomial time reduction form the graph bisection problem to $\Delta-2$BGP. Also, the decision version is used, that is, given a graph G and a positive integer B, to determine whether there is a 2-balanced partitioning solution such that the partitioning cost is not larger than B. Then, given an instance (G, B) of the graph bisection problem, the reduction will compute an instance (H, C) of $\Delta-2$BGP. The details of the reduction can be explained as follows. (1) First, an even integer X is computed, such that X satisfies $X \geq 2 \cdot \deg(G)$ and $X \geq |V_G| + 1$ where $\deg(G)$ is the maximum degree of the graph G. (2) Then, as shown in Fig. 3(a), for each node $v_i \in G$, a complete graph (or clique) C_i is built such that C_i includes X nodes and there is an edge between any two nodes. Let H be the graph composed of all C_i. (3) Next, several triangles are added into H, and the whole procedure will guarantee that each node will be utilized at most once. For each edge $(v_i, v_j) \in G$, assuming that $i < j$, randomly choose two unused nodes c_{i1} and c_{i2} in C_i and one unused node c_j in C_j, and add two edges (c_{i1}, c_j) and (c_{i2}, c_j) to H. As shown in Fig. 3, assume that the input graph G for the bisection problem is shown in Fig. 3(a), the result graph H constructed by the reduction for the $\Delta-2$BGP problem is shown in Fig. 3(b). (4) Finally, let the parameter C in $\Delta-2$BGP be equal to the parameter B in the bisection problem.

Obviously, the above reduction can be finished within $O(|E_G| \cdot X + |V_G| \cdot X^2)$ time, which can be guaranteed by setting X to be the smallest even number satisfying the conditions, that is $X = O(poly(|G|))$. Then, the correctness of the reduction can be verified by the following two aspects.

$\Rightarrow$ If there is a 2-balanced partitioning solution $\text{PAR}_G = \{S_G, T_G\}$ for G, such that the corresponding cost is not larger than B, we can construct a 2-balanced partitioning solution $\text{PAR}_H = \{S_H, T_H\}$ for H satisfying $\text{cost}_\Delta(\text{PAR}_H) \leq C = B$. For each node $v_i \in S_G$, the construction will add all nodes related with C_i in H to S_H, and for each node $v_j \in T_G$, the nodes of C_j in H will be added into T_H. It is not hard to verify that the partitioning solution defined by PAR_H will not destroy any triangle in C_i but only may destroy the triangles built in the third step of the reduction. For that step, the nodes utilized are will not be reused during the construction, so the obtained triangles will not connect with each other. Meanwhile, if some edge (v_i, v_j) of G is cut by PAR_G, then the corresponding triangle in H must be destroyed also by PAR_H, and vice versa. Therefore, we have $\text{cost}_\Delta(\text{PAR}_H) \leq C = B$.

$\Leftarrow$ If there is a 2-balanced partitioning solution $\text{PAR}_H = \{S_H, T_H\}$ for H and the cost is no larger than C, then a 2-balanced partitioning solution $\text{PAR}_G = \{S_G, T_G\}$ for G satisfying $\text{cost}(\text{PAR}_G) \leq B = C$ can be constructed. First, if $C \geq \frac{|V_G|(|V_G|-1)}{2}$, because $|E_G| \leq \frac{|V_G|(|V_G|-1)}{2}$, then G has a trivial 2-balanced partitioning solution since any solution will not cause a cost larger than C. Therefore, only the case when $C < \frac{|V_G|(|V_G|-1)}{2}$ is considered in the followings.

In this case, it can be verified that, given an arbitrary C_i in H, the nodes in C_i will all belong to S_H or T_H, that is the solution PAR_H will not split C_i. It can be proved by contradiction. Let us assume that PAR_H split C_i. Then, consider how many triangles in C_i are destroyed. There are at least $|V_G| + 1$ nodes totally in C_i, and the two parts obtained by PAR_H are of size x and y respectively. Without loss of generality, it is assumed that $x \leq y$. Then, it can be known that at least $F(x) = x \cdot \frac{y(y-1)}{2} = x \cdot \frac{(X-x)(X-1-x)}{2}$ triangles are destroyed, where $1 \leq x \leq X/2$. The relationship between x and the function value $F(x)$ is shown intuitively in Fig. 4. It is not hard to find that when the condition $1 \leq x \leq X/2$ is satisfied, the two extreme values of $F(x)$ are $X^3/8 - X^2/4$ (for $x = X/2$) and $(X-1)(X-2)/2$ (for $x = 1$). Furthermore, when $X \geq 2$ which can be satisfied during the construction of the reduction, the minimum value is obtained by taking $x = 1$. According to the reduction, we have $X \geq |V_G| + 1$, therefore, $F(1) = (X-1)(X-2)/2 \geq (|V_G|-1)|V_G|/2$. As
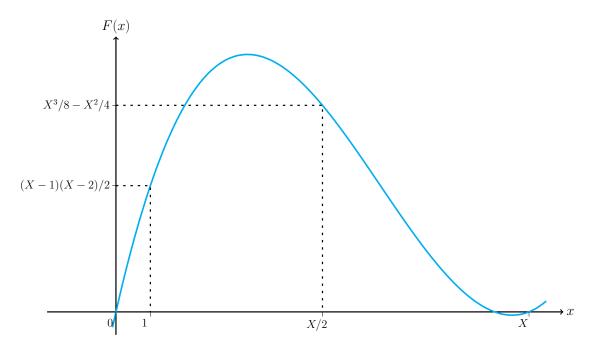
Fig. 4. An Illustration Example of $F(x)$ used in the proof of Theorem 7

a consequence, we have $\mathsf{cost}_\Delta(\mathrm{PAR}_H) \geq F(1) \geq (|V_G| - 1)|V_G|/2 > C$, which is a contradiction. Therefore, it can be known that PAR_H will not split C_i, that is all nodes of C_i belong to S_H or T_H.

Next, we can use the partitioning solution for H to build a 2-balanced partitioning solution for G, whose details are as follows. For each clique C_i of H, if all nodes of C_i belong to S_H, the related node v_i of G will be placed into S_G. Otherwise, the nodes of C_i belong to T_H and the corresponding node will be placed into T_G. Then, it can be verified that some edge (v_i, v_j) in G is cut by the partitioning solution PAR_G if and only if the triangle constructed for the edge (v_i, v_j) in H is destroyed by PAR_H. Thus, we have $\mathsf{cost}(\mathrm{PAR}_G) = \mathsf{cost}_\Delta(\mathrm{PAR}_H) \leq C = B$.

In conclusion, there is a polynomial time reduction from the graph bisection problem to the $\Delta-2\mathrm{BGP}$ problem, and the $\Delta-2\mathrm{BGP}$ problem is NP-hard. Finally, it is proved that the $\Delta-2\mathrm{BGP}$ problem is NP-complete. □

According to Theorem 7, the proof of Theorem 6 can be given as follows.

Proof (The brief proof of Theorem 6). The lower bound for the MkBGP can be obtained trivially, since, according to Theorem 7, when $k = 2$ the M2BGP problem is NP-hard, and it is a special case of MkBGP.

Therefore, in the followings, all we need to do is to explain that MkBGP belongs to NP. The corresponding decision variant of MkBGP can be expressed as follows. Given a graph G, a positive integer k, a c-Motif M and a positive integer C, the problem is to determine whether there is a k-balanced partitioning solution PAR_G for G such that $\mathsf{cost}_M(\mathrm{PAR}_G) \leq C$. Then, a NP algorithm for deciding the MkBGP problem can be designed as follows. (1) First, guess a k-balanced partitioning solution PAR_G for G and initialize a counter count using 0. (2) Then, for each pair $(u_1, \cdots, u_c) \in V_G^c$, if the induced subgraph obtained on $(u_1, \cdots, u_c)$ contains the same structure as M, the process will continue, otherwise, the following steps will be ingored. Next, it will be checked whether all $\mathrm{PAR}_G(u_i)$ ($i \in [1, c]$) are same, and the counter count will be increased by 1 if not same. (3) Finally, if count $\leq C$, the answer 'yes' will be returned, otherwise, the answer 'no' will be returned.

The correctness of the above algorithm can be verified easily. Since all operations can be implemented within $O(|V_G|^c \cdot c)$ time, the above algorithm is a NP algorithm for MkBGP. Thus, MkBGP is in NP.

13

Finally, it has been proved that the MkBGP problem is NP-complete, even for the special case of $k = 2$. $\square$

4.2 Analysis of Inapproximabilities

According to the above results, it can be known that, unless P = NP, it is impossible to design polynomial time algorithm for MkBGP. The impossibility remains the same even when $k = 2$ and M is a triangle. Then, a natural direction is trying to design efficient approximation algorithms for MkBGP.

Theorem 8. *Assume that P $\neq$ NP, even if the motif M is a triangle, there is no polynomial time algorithms for the MkBGP problem with finite approximation ratio.* $\square$

Proof. The proof is similar with [2], which will build a reduction from the 3-Partition problem to MkBGP. The input of the 3-Partition problem is composed of $n = 3k$ positive integers $\{a_1, \cdots, a_n\}$ and a threshold S satisfying $S/4 < a_i < S/2$ and $\sum_{i=1}^{n} a_i = kS$. The 3-Partition problem is to determine whether there is a partitioning method to group each three integers and the corresponding sum is exactly S. It is also one of the most classical NP-complete problems[11]. Fore more, the 3-partition problem is strongly NP-complete, that is, even if the time cost of the algorithm can be measured using both the input size n and the maximal integer v_{max} as the parameters, the problem still does not admit polynomial time algorithms unless P = NP.

Using the unary representation of the 3-Partition problem, the following reduction can compute an instance of MkBGP within polynomial time. For each integer a_i, construct a clique K_i of size a_i, the union of all K_is is just the input graph H of MkBGP. Then, it can be proved that, the answer of the 3-Partition problem is 'yes' if and only if the reduced MkBGP problem admits a k-balanced partitioning solution with cost 0. Assume that there is a polynomial time algorithm $\mathcal{A}$ for MkBGP with finite approximation ratio, then we build the input H and invoke the algorithm $\mathcal{A}$ using parameter k. Let the output of $\mathcal{A}$ be PAR_H. If $\mathsf{cost}_\triangle(\mathrm{PAR}_H) = 0$, there is a 0-cost partitioning solution for H and then the answer of the original 3-Partition problem is 'yes'. Otherwise, if $\mathsf{cost}_\triangle(\mathrm{PAR}_H) > 0$, since the approximation ratio of $\mathcal{A}$ is finite, we have $\mathsf{cost}_\triangle(\mathrm{PAR}_{opt}) > 0$, that is there is no 0-cost partitioning solution for H and the answer of the 3-Partition problem should be 'no'. Combining the reduction algorithm with $\mathcal{A}$, we can obtain an algorithm $\mathcal{B}$, and $\mathcal{B}$ is a pseudo polynomial time algorithm for 3-Partition, which is a contradiction since it is strongly NP-complete. Finally, the proof is finished. $\square$

5 Graph Partitioning Algorithm for Optimizing Motif Computation

5.1 SDP Based Approximation Algorithm for $\Delta - k$BGP

First, focusing on the special case that the motif is a triangle, an approximation algorithm based on SDP is designed for the MkBGP problem, and then the proposed algorithm is extended to the general case.

The Semidefinite Program for MkBGP Given the input graph G and an integer k, the semidefinite program for the $\Delta - k$BGP problem can be expressed by Equation (10).

$$\min \sum_{\triangle_{\langle a,b,c \rangle} \in G} \|\boldsymbol{x}_{abc}\|_2^2 \tag{10}$$

$$\text{s.t.} \quad \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \|\boldsymbol{v} - \boldsymbol{w}\|_2^2 \geq \|\boldsymbol{u} - \boldsymbol{w}\|_2^2 \quad \forall u, v, w \in V \tag{11}$$

$$\sum_{v \in S} \frac{1}{2} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \geq |S| - \frac{n}{k} \qquad \forall S \subseteq V, u \in S \tag{12}$$

$$\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2} \|\boldsymbol{c} - \boldsymbol{a}\|_2^2$$

$$\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2} \|\boldsymbol{a} - \boldsymbol{b}\|_2^2$$

$$\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2} \|\boldsymbol{b} - \boldsymbol{c}\|_2^2 \qquad \forall \triangle_{\langle a,b,c \rangle} \in G \tag{13}$$

Here, similar with Equation (5), the $\|\cdot\|_2^2$ in Equation (10) is the $L2$ normal form. The constraints represented by Equation (11) and (12) are the triangle inequality defined over ℓ_2^2 and the spreading constraints. The principle of using vectors to represent the partitioning information is also similar with Equation (5). If the distances between $\boldsymbol{u}$ and $\boldsymbol{v}$ is too large, it means that $\mathrm{PAR}(u) \neq \mathrm{PAR}(v)$, otherwise, $\mathrm{PAR}(u) = \mathrm{PAR}(v)$. The goal of the SDP shown in Equation (10) is to simulate the cost function $\mathsf{cost}_{M=\Delta}(\cdot)$ defined by Equation (3) for the MkBGP problem, which can be explained as follows. For each triangle $\Delta_{\langle a,b,c\rangle}$ in G, the value of the corresponding vector $\boldsymbol{x}_{abc}$ indicates whether the triangle $\Delta_{\langle a,b,c\rangle}$ is destroyed in the partitioning solution. According to the definition of $\mathsf{cost}_M(\cdot)$ in MkBGP, the triangle $\Delta_{\langle a,b,c\rangle}$ is destroyed if and only if at least one edge in the triangle is spitted by the partition. Intuitively, for Equation (10), the length $\|\boldsymbol{x}_{abc}\|_2^2$ of the vector $\boldsymbol{x}_{abc}$ is expected to be 0 or 1 in the ideal case. If $\|\boldsymbol{x}_{abc}\|_2^2 = 1$, the partitioning solution corresponding to the SDP solution will destroy the triangle $\Delta_{\langle a,b,c\rangle}$, otherwise, all nodes of the triangle $\Delta_{\langle a,b,c\rangle}$ are placed in one partition. In the practical setting, when building the partitioning solution based on SDP, if $\|\boldsymbol{x}_{abc}\|_2^2$ is small, it will tend to put all the three nodes in one partition, and only when the value of $\|\boldsymbol{x}_{abc}\|_2^2$ is not small it will be possible to split the three nodes. For constructing the relationship between splitting triangles and the vectors, the constraints shown in Equation (13) are added, where three constraints for each triangle are added and the total size can be bounded by $O(n^3)$. According to the discussions above, the expression like $\frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2$ can be used to determine whether two nodes should be put together. Then, according to the constraint $\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2$ defined by Equation (13), if $\frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2$ takes a rather large value, the value of $\|\boldsymbol{x}_{abc}\|_2^2$ is large too. Similarly, only when $\frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2 = 0$, the value of $\|\boldsymbol{x}_{abc}\|_2^2$ can be 0 and the optimizing goal (10) can be minimized then. Therefore, because the optimizing goal (10) should be minimized, for the three constraints described in Equation (13), the value of $\|\boldsymbol{x}_{abc}\|_2^2$ is 0, if and only if the three constraints satisfy that the values of $\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2}\|\boldsymbol{c} - \boldsymbol{a}\|_2^2$, $\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|_2^2$ and $\|\boldsymbol{x}_{abc}\|_2^2 \geq \frac{1}{2}\|\boldsymbol{b} - \boldsymbol{c}\|_2^2$ are all 0. That is, the triangle $\Delta_{\langle a,b,c\rangle}$ is not destroyed. Otherwise, the value of $\|\boldsymbol{x}_{abc}\|_2^2$ will be at least no less than the max value in the three constraints.

Because the size of the constraints defined by Equation (13) can be bounded by $O(n^3)$, then, according to Theorem 3, it can be proved that the SDP (10) can be solved within polynomial time.

Proposition 3. *The SDP shown in Equation* (10) *can be optimized in polynomial time.* □

The Tri-kBGPartition Algorithm In this part, the Tri-kBGPartition Algorithm for solving MkBGP is introduced, whose main idea is to first solve the corresponding SDP and then use the rounding techniques to construct the partitioning solution.

The details of Tri-kBGPartition are shown in Algorithm 2. First, Tri-kBGPartition needs to process all triangle structures in G and maintain the result in the set S_Δ (line 2). According to Lemma **??**, Tri-kBGPartition can solve the SDP shown in Equation (10) within polynomial time by invoking a sophisticated solver (line 3), and let the optimized solution obtained be $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$. Then, the set V is used to initialize S and the partition result PAR_G is set to be empty (line 4). Next, using the rounding technique of [14], the partitioning solution can be built as follows.

✓ First, Tri-kBGPartition builds a vector $\boldsymbol{r}$ randomly and α_{Ck} based on a variable X generated by a normal distribution $N(0, 1)$, where the value of α_m satisfies $\Pr[X \geq \alpha_m] = \frac{1}{m}$ (line 6).

✓ Then, Tri-kBGPartition utilizes a function $g(\cdot)$ to transform the ℓ_2^2 distance to a function on ℓ_2 (line 7). The transformation satisfies the following conditions. As shown by [5], given n vectors which satisfy the triangle inequality defined on ℓ_2^2, there exists a constant $\delta, A > 0$ and for any $\Lambda > 0$ we can find a function $g_\Lambda : \mathbb{R}^m \to \mathbb{R}^n$ such taht for any two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we have (i) $\|g_\Lambda(\boldsymbol{u}) - g_\Lambda(\boldsymbol{v})\|_2 \leq \frac{A\sqrt{\log n}}{\Lambda}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$, (ii) $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \geq \Lambda \Rightarrow \|g_\Lambda(\boldsymbol{u}) - g_\Lambda(\boldsymbol{v})\|_2 \geq \delta$ and (iii) $\|g_\Lambda(\boldsymbol{u})\|_2 = 1$. Here, the function $g(\cdot)$ used by Tri-kBGPartition is a special case of $g_\Lambda(\cdot)$ when $\Lambda = 2/3$.

✓ Using the above two techniques, in each iteration, Tri-kBGPartition checks whether the size of S is larger than the $\frac{2n}{k}$ (line 8). If the size of S is still too large, Tri-kBGPartition randomly selects a vector $\boldsymbol{r}$, and for each vector $\boldsymbol{v}_i$ in S, compute the inner product of $\boldsymbol{r}$ and $g(\boldsymbol{v}_i)$. If the result is larger than α_{Ck}, the node v_i will be added into S_r.

Algorithm 2: TRI-kBGPARTITION (Triangle Computing based k Balanced Graph Partitioning)

Input: A Graph $G = \{V, E\}$ and an integer k.
Output: A partition $\text{PAR}_G = \{V_1, V_2, \cdots, V_k\}$ for G.

1 **function** TRI-kBGPARTITION(G, k)
2   Compute and maintain all triangles of G into S_Δ;
3   Solve sdp (10) and let $opt = \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$ be the optimal solution;
4     $S \leftarrow V$, $\text{PAR}_G \leftarrow \emptyset$;
5   **while** $|S| > \frac{2n}{k}$ **do**
6   Randomly choose $\boldsymbol{r} \in \mathbb{R}^m$ s.t. $r_i \sim N(0, 1)$ and all r_is are i.i.d.;
7         $S_r \leftarrow \{v_i \mid v_i \in S, g(\boldsymbol{v}_i) \cdot \boldsymbol{r} \geq \alpha_{Ck}\}$;
8   **if** $0 < |S_r| \leq \frac{2n}{k}$ **then**
9   Add S_r to PAR_G;
10            $S \leftarrow S \setminus S_r$;

11   **while** $|\text{PAR}_G| > k$ **do**
12   Find the smallest two partition items V' and V'' of PAR_G;
13   Remove V' and V'' from PAR_G;
14   Add $V' \cup V''$ to PAR_G;

15   **return** PAR_G;

✓ During each iteration, if the size of the selected S_r is no larger than $\frac{2n}{k}$, it will be treated as a feasible partition, and the set S and PAR_G will be updated (line 8-10). Obviously, if each iteration can extract one feasible partition, then the iterations defined between line 6 and 10 of TRI-kBGPARTITION can be terminated after being executed for polynomial times. However, the worst case is that the size of S_r selected in each iteration is too large, in this case, the running time can not be bounded efficiently. Therefore, only the expected time of TRI-kBGPARTITION is bounded in the following analysis. According to [14], the probability that the size of S_r is too large will be not larger than $1/2$. Therefore, the expected time of TRI-kBGPARTITION is polynomial.

After the rounding procedure, in fact, TRI-kBGPARTITION can not always guarantee that there are exactly k partitions obtained. Therefore, TRI-kBGPARTITION needs a post-process to compute the final k partitions (line 11-14). If the number of partitions is larger than k, TRI-kBGPARTITION selects the smallest two partitions V' and V'' from PAR_G, and merges them into one partition. It is not hard to verify that the size of the merged partition will not be larger than $\frac{2n}{k}$. TRI-kBGPARTITION repeats the above procedure on PAR_G until $|\text{PAR}_G| = k$. Finally, the result PAR_G will be returned by TRI-kBGPARTITION (line 15).

Approximation Ratio of the Tri-kBGPartition Algorithm In this part, the approximation ratio of TRI-kBGPARTITION is analyzed, which will study the relationship between the optimized solutions of SDP and MkBGP first, and then analyze the ratio between the result returned by TRI-kBGPARTITION and the optimized solution of SDP.

Theorem 9. *Given an input graph G and an integer k, for the $\Delta - kBGP$ problem, the optimal solution of the SDP shown in (10) is a lower bound of the optimal partitioning cost of $\Delta - kBGP$.* □

Proof. The details are omitted due to the space limits.

According to [14], we have the following result.

Lemma 2 ([14]). *Given two nodes u and v in G, the probability that TRI-kBGPARTITION will place them to two different partitions is not larger than $3AB\sqrt{2\log(2Ck)\log n} \cdot \|\boldsymbol{u} - \boldsymbol{v}\|_2^2$, where the parameters A, B and C are constants whose details can be found in [14].* □

Given a triangle $\Delta_{\langle a,b,c \rangle}$ in G, we use the notation $\|\Delta_{\langle a,b,c \rangle}\|_2^2$ to represent the value of $\|\boldsymbol{a} - \boldsymbol{b}\|_2^2 + \|\boldsymbol{b} - \boldsymbol{c}\|_2^2 + \|\boldsymbol{a} - \boldsymbol{c}\|_2^2$. Let $\mathcal{A}_{abc}$ be the event that the triangle $\Delta_{\langle a,b,c \rangle}$ is destroyed by TRI-kBGPARTITION. By analyzing the probability of $\mathcal{A}_{abc}$, we have the following result.

Theorem 10. *For a node subset $U \subseteq V$, which satisfies that the three nodes of $\Delta_{\langle a,b,c \rangle}$ are all in U, assuming that during some iteration of* TRI-kBGPARTITION *we have $S = U$, the probability that the event $\mathcal{A}_{abc}$ happens satisfies the following condition.*

$$\Pr[\mathcal{A}_{abc}|S = U] \leq \frac{2\|\Delta_{\langle a,b,c \rangle}\|_2^2}{3} \cdot 3AB\sqrt{2\log(2Ck)\log n} \tag{14}$$

Here, the parameters A, B and C are same as Lemma 2. □

Proof. Let $\mathcal{A}_{x,y}$ be the event that x and y are separated. Assume that $\|a - b\|_2^2 \leq \|b - c\|_2^2 \leq \|a - c\|_2^2$. Obviously, the probability of $\mathcal{A}_{abc}$ can be decomposed into two parts as follows.

$$\Pr[\mathcal{A}_{abc}] = \Pr[\mathcal{A}_{a,b}] + \Pr[\overline{\mathcal{A}}_{a,b} \wedge \mathcal{A}_{b,c}]$$

Here, $\overline{\mathcal{A}}$ is the event that $\mathcal{A}$ does not happen. Then, we have the following result.

$$\begin{aligned}
\Pr[\mathcal{A}_{abc}] &= \Pr[\mathcal{A}_{a,b}] + \Pr[\overline{\mathcal{A}}_{a,b} \wedge \mathcal{A}_{b,c}] \\
&\leq \Pr[\mathcal{A}_{a,b}] + \Pr[\mathcal{A}_{b,c}] \\
&\leq \frac{(1+\epsilon)AB}{\epsilon}\sqrt{2\log(Ck/\epsilon)\log n}\left(\|a-b\|_2^2 + \|b-c\|_2^2\right) \\
&\leq \frac{2}{3}\frac{(1+\epsilon)AB}{\epsilon}\sqrt{2\log(Ck/\epsilon)\log n}\left(\|a-b\|_2^2 + \|b-c\|_2^2 + \|a-c\|_2^2\right) \\
&= \frac{2\|\Delta_{\langle a,b,c \rangle}\|_2^2}{3} \cdot \frac{(1+\epsilon)AB}{\epsilon}\sqrt{2\log(Ck/\epsilon)\log n}
\end{aligned}$$

Here, the second inequality is because of Lemma 2, the third inequality is derived from the fact that the distance between a and c is maximized in the triangle $\triangle_{\langle a,b,c \rangle}$. □

Based on Theorem 9 and 10, the approximation ratio of TRI-kBGPARTITION can be obtained by the following theorem.

Theorem 11 (Approximation Ratio of Tri-kBGPartition). *The triangle based balanced graph partitioning problem can be solved by a* bi-criteria *approximation algorithm within expected ratio $O(\sqrt{\log k \log n})$, where the size of each partition is at most $\frac{2n}{k}$.* □

Proof. Given an instance $I = \langle G, k \rangle$ of $\Delta - k$BGP, let PAR_G be the partitions built by Algorithm 2 and PAR_G^* be the k-balanced partition of I with the optimal partitioning cost. For more, let $\mathsf{cost}_\Delta(\mathrm{PAR}_G)$ and $\mathsf{cost}_\Delta(\mathrm{PAR}_G^*)$ be the corresponding partitioning cost of PAR_G and PAR_G^*.

Since PAR_G^* is optimal, we have $\mathsf{cost}_\Delta(\mathrm{PAR}_G) \geq \mathsf{cost}_\Delta(\mathrm{PAR}_G^*)$ obviously. Then, since the partitioning solution of TRI-kBGPARTITION is based on the optimal solution of the SDP shown in (10), let the optimal solution of the SDP be S^*. Then, according to Theorem 9, we have $S^* \leq \mathsf{cost}_\Delta(\mathrm{PAR}_G^*)$. Therefore, $S^* \leq \mathsf{cost}_\Delta(\mathrm{PAR}_G^*) \leq \mathsf{cost}_\Delta(\mathrm{PAR}_G)$.

Because the rounding techniques used by TRI-κBGPARTITION is randomized, we analyze the expected cost of the partitioning solution returned by TRI-κBGPARTITION.

$$\mathbf{E}[\mathsf{cost}_\Delta(\mathrm{PAR}_G)] = \sum_{\Delta_{\langle a,b,c\rangle}\in G} \left(\Pr[\mathcal{A}_{abc}] \right)$$

$$\leq \sum_{\Delta_{\langle a,b,c\rangle}\in G} \left(\frac{2\|\Delta_{\langle a,b,c\rangle}\|_2^2}{3} \cdot 3AB\sqrt{2\log(2Ck)\log n} \right)$$

$$\leq \sum_{\Delta_{\langle a,b,c\rangle}\in G} \left(4\|\boldsymbol{x}_{abc}\|_2^2 \cdot 3AB\sqrt{2\log(2Ck)\log n} \right)$$

$$= \left(12AB\sqrt{2\log(2Ck)\log n} \right) \cdot S^*$$

Here, the first inequality is based on Theorem 10, the second one is based on the constraints defined by (13), and the last one is obtained by considering the definition of the optimizing goal of (10).

Thus, we have the following result.

$$S^* \leq \mathsf{cost}_\Delta(\mathrm{PAR}_G^*) \leq \mathbf{E}[\mathsf{cost}_\Delta(\mathrm{PAR}_G)] \leq \left(12AB\sqrt{2\log(2Ck)\log n} \right) \cdot S^*$$

Finally, we can obtain the followings.

$$\mathbf{E}\left[\frac{\mathsf{cost}_\Delta(\mathrm{PAR}_G)}{\mathsf{cost}_\Delta(\mathrm{PAR}_G^*)} \right] \leq 12AB\sqrt{2\log(2Ck)\log n} = O\left(\sqrt{\log k \log n} \right)$$

That is, the expected approximation ratio of TRI-κBGPARTITION can be bounded by $O(\sqrt{\log k \log n})$.

□

5.2 Extension to the General MkBGP Problem

Based on the algorithm designed for the $\Delta-k$BGP problem, for the general case of MkBGP, TRI-κBGPARTITION can be naturally extended and guarantee the same approximation ratio. Here, the corresponding SDP can be represented by the following form.

$$\min \quad \sum_{H\in G\langle M\rangle} \|\boldsymbol{x}_H\|_2^2 \tag{15}$$

$$\text{s.t.} \quad \|\boldsymbol{u}-\boldsymbol{v}\|_2^2 + \|\boldsymbol{v}-\boldsymbol{w}\|_2^2 \geq \|\boldsymbol{u}-\boldsymbol{w}\|_2^2 \quad \forall u,v,w\in V \tag{16}$$

$$\sum_{v\in S} \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 \geq |S|-\frac{n}{k} \qquad \forall S\subseteq V, u\in S \tag{17}$$

$$\|\boldsymbol{x}_H\|_2^2 \geq \frac{1}{2}\|\boldsymbol{a}-\boldsymbol{b}\|_2^2 \qquad \forall H\in G\langle M\rangle, (a,b)\in H \tag{18}$$

Here, the first two constraints used in (15) are similarly defined as previous. The second constraints shown in (18) is an extension of the corresponding constraints used by the $\Delta-k$BGP problem. If M is a motif of size c, then for each $H\in G\langle M\rangle$, the SDP will includes c conditions.

Obviously, when the size of M is c, compared with (10), the SDP shown in (15) needs $|V|^c$ constraints in addition. According to Theorem 3, it can be still verified that the SDP shown in (15) can be solved within polynomial time.

Finally, the proof of Theorem 9 and 11 can be extended to the general case also, and we have the following result.

Theorem 12. *Given a c-motif M, there is a bi-criteria $(k,2)$ approximation algorithm with polynomial time cost for MkBGP, where the expected approximation ratio can be bounded by $O(\sqrt{\log k \log n})$ and the size of each output partition is at most $\frac{2n}{k}$.* □

18

6 Conclusion

In this paper, two typical problems of graph partitioning motivated by the big data computing applications are studied. For the WkBGP problem, a bi-criteria polynomial time algorithm with $O(\sqrt{\log n \log k})$ approximation ratio is designed. For the MkBGP problem, it is proved to be NP-complete and impossible to be approximated within finite ratio, then a bi-criteria polynomial time algorithm with $O(\sqrt{\log n \log k})$ approximation ratio is designed.

References

1. Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.
2. Konstantin Andreev and Harald Räcke. Balanced graph partitioning. *Theory Comput. Syst.*, 39(6):929–939, 2006.
3. Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):5:1–5:37, April 2009.
4. Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 222–231. ACM, 2004.
5. Eden Chlamtac, Konstantin Makarychev, and Yury Makarychev. How to play unique games using embeddings. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), Berkeley, California, USA*, pages 687–696, 2006.
6. Dingzhu Du, Keyi Ge, and Xiaodong Hu. *Design and Analysis of Approximation Algorithms*. Springer, New York, 2011.
7. Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Fast approximate graph partitioning algorithms. *SIAM J. Comput.*, 28(6):2187–2214, 1999.
8. Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000.
9. Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Xiaojian Luo, Ruiqi Xu, Qiang Yin, Wenyuan Yu, and Jingren Zhou. Application driven graph partitioning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD'20, pages 1765–1779, New York, NY, USA, 2020. Association for Computing Machinery.
10. Andreas Emil Feldmann and Luca Foschini. Balanced partitions of trees and applications. *Algorithmica*, 71(2):354–376, 2015.
11. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, USA, 1979.
12. M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.
13. Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In Chandu Thekkath and Amin Vahdat, editors, *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8-10, 2012*, pages 17–30. USENIX Association, 2012.
14. Robert Krauthgamer, Joseph (Seffi) Naor, and Roy Schwartz. Partitioning graphs into balanced components. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'09, pages 942–949, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.
15. Claudio Martella, Dionysios Logothetis, Andreas Loukas, and Georgos Siganos. Spinner: Scalable graph partitioning in the cloud. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 1083–1094. IEEE Computer Society, 2017.
16. Anil Pacaci and M. Tamer Özsu. Experimental analysis of streaming algorithms for graph partitioning. In *Proceedings of the 2019 ACM SIGMOD International Conference on Management of Data*, SIGMOD'19, pages 1375–1392, New York, NY, USA, 2019. ACM.
17. Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264. ACM, 2008.
18. Horst D. Simon and Shang-Hua Teng. How good is recursive bisection? *SIAM J. Sci. Comput.*, 18(5):1436–1445, 1997.