Journal of Combinatorial Optimization (2023) 45:127
https://doi.org/10.1007/s10878-023-01058-x

Balanced connected partitions of graphs: approximation, parameterization and lower bounds

Phablo F. S. Moura[1] · **Matheus J. Ota**[2] · **Yoshiko Wakabayashi**[3]

Accepted: 8 June 2023

Abstract
A connected k-partition of a graph is a partition of its vertex set into k classes such that each class induces a connected subgraph. Finding a connected k-partition in which the classes have similar size is a classical problem that has been investigated since late seventies. We consider a more general setting in which the input graph $G = (V, E)$ has a nonnegative weight assigned to each vertex, and the aim is to find a connected k-partition in which every class has roughly the same weight. In this case, we may either maximize the weight of a lightest class (MAX–MIN BCP$_k$) or minimize the weight of a heaviest class (MIN–MAX BCP$_k$). Both problems are NP-hard for any fixed $k \geq 2$, and equivalent only when $k = 2$. In this work, we propose a simple pseudo-polynomial $\frac{3}{2}$-approximation algorithm for MIN–MAX BCP$_3$, which is an $\mathcal{O}(|V||E|)$ time $\frac{3}{2}$-approximation for the unweighted version of the problem. We show that, using a scaling technique, this algorithm can be turned into a polynomial-time $(\frac{3}{2} + \varepsilon)$-approximation for the weighted version of the problem with running-time $\mathcal{O}(|V|^3|E|/\varepsilon)$, for any fixed $\varepsilon > 0$. This algorithm is then used to obtain, for MIN–MAX BCP$_k$, $k \geq 4$, analogous results with approximation ratio $(\frac{k}{2} + \varepsilon)$. For $k \in \{4, 5\}$, we are not aware of algorithms with approximation ratios better than those. We also consider fractional bipartitions that lead to a unified approach to design simpler approximations for both MIN–MAX and MAX–MIN versions. Additionally, we propose a fixed-parameter tractable algorithm based on integer linear programming for the unweighted MAX–MIN BCP parameterized by the size of a vertex cover. Assuming the

✉ Phablo F. S. Moura
phablo.moura@kuleuven.be

Matheus J. Ota
mjota@uwaterloo.ca

Yoshiko Wakabayashi
yw@ime.usp.br

1 Research Center for Operations Research & Statistics, KU Leuven, Leuven, Belgium

2 Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada

3 Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, Brazil

Published online: 20 June 2023

Exponential-Time Hypothesis, we show that there is no subexponential-time algorithm to solve the MAX–MIN and MIN–MAX versions of the problem.

Keywords Balanced connected partition · Fractional partition · Approximation algorithms · Fixed parameter tractable · Complexity lower bound

Mathematics Subject Classification 68Q25 · 68W25 · 05C85

1 Introduction

The problem of partitioning a connected graph into a given number $k \geq 2$ of connected subgraphs with prescribed orders was first studied by Lovász (1977) and Győri (1978) in the late seventies. Let $[k]$ denote the set $\{1, 2, \dots, k\}$, for every integer $k \geq 1$. A *connected k-partition* of a connected graph $G = (V, E)$ is a partition of V into nonempty classes $\{V_i\}_{i=1}^{k}$ such that, for each $i \in [k]$, the subgraph $G[V_i]$ is connected, where $G[V_i]$ denotes the subgraph of G induced by the set of vertices V_i.

We denote by (G, w) a pair consisting of a connected graph $G = (V, E)$ and a function $w \colon V \to \mathbb{Q}_{\geq}$ that assigns nonnegative weights to the vertices of G. For each $V' \subseteq V$, we define $w(V') = \sum_{v \in V'} w(v)$. Furthermore, if $G' = (V', E')$ is a subgraph of G, we write $w(G')$ instead of $w(V')$. If $\mathcal{P} = \{V_i\}_{i \in [k]}$ is a connected k-partition of G, then $w^+(\mathcal{P})$ stands for $\max_{i \in [k]} \{w(V_i)\}$, and $w^-(\mathcal{P})$ stands for $\min_{i \in [k]} \{w(V_i)\}$.

The concept of balance of the classes of a connected partition can be expressed in different ways. In this work, we consider two related variants whose objective functions express this concept.

Problem Min-Max Balanced Connected k-Partition (MIN−MAX BCP$_k$)
INSTANCE: a connected graph $G = (V, E)$, and a weight function $w \colon V \to \mathbb{Q}_{\geq}$.
FIND: a connected k-partition $\mathcal{P}$ of G.
GOAL: minimize $w^+(\mathcal{P})$.

The problem MAX−MIN BCP$_k$ is defined analogously: it has the same set of instances, but it seeks a connected k-partition $\mathcal{P}$ that maximizes $w^-(\mathcal{P})$.

The problems MIN−MAX BCP$_2$ and MAX−MIN BCP$_2$ are equivalent, that is, an optimal solution for one of the versions is also an optimal solution for the other version (but they may have different optimal values). However, for $k > 2$, both problems are not equivalent.

For all problems mentioned here the input graph G is always simple and connected (and possibly with further properties). We also use the convention that n (resp. m) is the number of vertices (resp. edges) of the graph under consideration.

Throughout this paper we assume that $k \geq 2$. When k is in the name of the problem, we are considering that k is fixed. The problems in which k is part of the instance are denoted similarly but without specifying k in the name (e.g. MAX−MIN BCP). The unweighted (or cardinality) versions of the problems refer to the case in which all vertices have equal weights, which may be assumed to be 1. We denote the corresponding problems as 1-MIN-MAX BCP$_k$, 1-MAX-MIN BCP$_k$, 1-MIN-MAX BCP and 1-MAX-MIN BCP.

In this paper, we show approximation algorithms for MIN−MAX BCP$_k$, but mention approximation results for both MIN−MAX BCP$_k$ and MAX−MIN BCP$_k$. We observe that whenever we refer to an approximation algorithm, we mean that it runs in polynomial time on the size of the instance. If an approximation ratio α can be guaranteed for an algorithm that may run in pseudo-polynomial time, we also refer to it as a *pseudo-polynomial α-approximation*. This is not a usual terminology, but it will be appropriate for our purposes.

Problems on balanced connected partitions can model a rich collection of applications in logistics, image processing, data base, operating systems, cluster analysis and robotics (Becker and Perl 1983; Lucertini et al. 1993; Maravalle et al. 1997; Zhou et al. 2019).

Dyer and Frieze (1985) proved that 1-MAX-MIN BCP$_k$ is NP-hard on bipartite graphs for every $k \geq 2$. Furthermore, 1-MAX-MIN BCP$_k$ has been shown by Chlebíková (1996) to be NP-hard to approximate within an absolute error guarantee of $n^{1-\varepsilon}$, for all $\varepsilon > 0$. For the weighted versions, Becker et al. (1998) proved that MAX−MIN BCP$_2$ is NP-hard on grid graphs. Wu (2012) showed that MAX−MIN BCP$_k$ is NP-hard on interval graphs for every k. Chataigner et al. (2007) proved that MAX−MIN BCP$_k$ is strongly NP-hard, even on k-connected graphs. Hence, unless P = NP, the problem MAX−MIN BCP$_k$ does not admit a fully polynomial-time approximation scheme (FPTAS). They also showed that, when k is part of the instance, the problem MAX−MIN BCP cannot be approximated within a ratio better than $6/5$, unless P = NP.

We now turn to existential or algorithmic results for the unweighted versions of the k-connected partition problems. When the input graph G is k-connected, Győri (1978), and Lovász (1977) proved that one can always find a connected k-partition where each class has a prescribed number of vertices (and each class contains a prescribed vertex). The proof of this result given by Győri (1978) does not seem to yield a polynomial-time algorithm for every k. Holyer (2019) presented a polynomial-time algorithm for $k = 2, 3, 4$. Before this result, Suzuki et al. (1990a) devised a linear-time algorithm to find a connected 2-partition on a 2-connected graph. When the input graph is 3-connected, Suzuki et al. (1990b) presented a quadratic-time algorithm to compute a connected 3-partition. If G is planar and 4-connected, Nakano et al. (1997) showed that a connected 4-partition can be found in linear time (also when the prescribed vertices are on the same face in some embedding of G).

For MAX−MIN BCP$_k$ (resp. MIN−MAX BCP$_k$), Perl and Schach (1981) (resp. Becker et al. (1982)) designed polynomial-time algorithm when the input graph is a tree. Also for trees, Frederickson (1991) proposed linear-time algorithms for both MAX−MIN BCP$_k$ and MIN−MAX BCP$_k$. Polynomial-time algorithms were also derived for MAX−MIN BCP$_2$ on graphs with at most two cut-vertices (Chlebíková 1996; Alimonti and Calamoneri 1999). For MAX−MIN BCP$_k$ on ladders, a polynomial-time algorithm was obtained by Becker et al. (2001).

Chlebíková (1996) designed a $(4/3)$-approximation algorithm for MAX- MIN BCP$_2$. Chen et al. (2020) observed that the algorithm obtained by Chlebíková has approximation ratio $5/4$ for MIN−MAX BCP$_2$ (but requires another analysis). These authors also obtained approximation algorithms with ratios $3/2$ and $5/3$ for MIN−MAX BCP$_3$ and MAX−MIN BCP$_3$, respectively. For the unweighted version 1- MIN- MAX BCP$_k$, $k \geq 3$, Chen et al. (2021) presented a $k/2$-approximation algorithm. In 2012, Wu (2012) designed a FPTAS for MAX−MIN BCP$_2$ restricted to interval graphs. When k is part of the input, recently Casel et al. (2021) derived a very involved 3-approximation algorithm for both MAX−MIN BCP and MIN−MAX BCP which is based on the crown decomposition of the graph. For recent exact algorithms based on mixed integer linear programs for these problems, we refer the reader to Miyazawa et al. (2020, 2021).

1.1 Contributions

We show an approximation algorithm for MIN−MAX BCP$_k$, $k \geq 3$, that was inspired by the $k/2$-approximation algorithm, designed by Chen et al. (2021) for (the unweighted version) 1- MIN- MAX BCP$_k$. The algorithm we present here has basically the same approximation ratio: namely, $k/2 + \varepsilon$, for any arbitrarily small $\varepsilon > 0$. When the weights assigned to the vertices of the input graph are bounded by a polynomial on the order of the graph, it achieves the ratio $k/2$. The additional constant ε in the ratio $k/2$ comes from a scaling technique used to deal with weights that might be very large. These results are presented in Sect. 2. We note that a $3/2$-approximation algorithm for MIN−MAX BCP$_3$ was obtained by Chen et al. (2020), but its analysis and implementation are slightly more complicated than the algorithm we show here.

For $k \geq 4$, we did not find in the literature approximation algorithms for MIN−MAX BCP$_k$. But when k is part of the input, the algorithm by Casel et al. (2021), mentioned above, is a 3-approximation for MIN−MAX BCP (and also for MAX−MIN BCP). Thus, only for $k \in \{4, 5\}$, the algorithm we show for MIN−MAX BCP$_k$ have the best ratio known in the literature. They are simple to be implemented.

We present results on a fractional connected 2-partition problem in Sect. 3. More precisely, given a fraction p/q such that $2/3 \leq p/q < 1$, we show an algorithm that produces either a connected bipartition $\{V_1, V_2\}$ of G with $w(V_1) \leq w(V_2) \leq \frac{p}{q} w(G)$, or reveals that G has a cut-vertex u and provides information on the weights of the components of $G - u$. Of course in this case, we do not have the notion of balance of the two classes. However, we shall see that such fractional bipartitions lead to a unified approach to design simpler approximation algorithms for both MIN−MAX and MAX−MIN versions of the problem.

In Sect. 4, we prove that 1-MAX-MIN BCP is fixed-parameter tractable when the parameter is the size of a vertex cover of the input graph. The proposed algorithm is based on an integer linear program that has a doubly exponential dependency on the size of a vertex cover. To the best of our knowledge, no FPT algorithm for balanced connected partition problems is described in the literature. We believe that the strategy used to model connected partitions may be useful to show that other problems involving connectivity constraints are fixed-parameter tractable when the parameter is the size of a vertex cover. On the negative side, we present lower bounds for solving the MAX−MIN or MIN−MAX version of the problem assuming the Exponential-Time

Hypothesis (ETH). More precisely, in Sect. 5, we show that there is no subexponential-time algorithm for solving any of these problems, unless ETH fails.

A preliminary version of this work containing parts of these results appeared in the Proceedings of the 8th Annual International Conference on Algorithms and Discrete Applied Mathematics (CALDAM 2022) (Moura et al. 2022). This paper contains full proofs and additional results: implementation details, fractional bipartition, and lower bounds.

2 Approximation algorithms for MIN−MAX BCP$_k$ with $k \geq 3$

Chen et al. (2021) devised an algorithm for 1-MIN-MAX BCP$_k$ with approximation ratio $\frac{k}{2}$. These authors show first an algorithm for 1- MIN- MAX BCP$_3$, and then derive algorithms for 1-MIN-MAX BCP$_k$, $k \geq 4$. The first algorithm iteratively applies two simple operations, called PULL and MERGE, to reduce (if possible) the size of the largest class (up to some bound). In what follows, we show how to generalize such operations for the *weighted* case to design a $(\frac{3}{2} + \varepsilon)$-approximation for MIN−MAX BCP$_3$, for any $\varepsilon > 0$. Then, we show how to use the connected 3-partition returned by this algorithm to obtain a connected k-partition for $k \geq 4$.

Throughout this section, k is a positive integer, and $(G = (V, E), w)$ denotes an instance of MIN−MAX BCP$_k$, where $w\colon V \to \mathbb{Q}_{\geq}$. When the pair (V, E) is not needed in the context, such an instance is denoted as (G, w). Also, when convenient, we denote by W the sum of the weights of the vertices in G, that is, $W = w(G)$. In this section, we assume without loss of generality that w is an integer-valued function (otherwise, we may simply multiply all weights by the least common multiple of the denominators).

The following trivial fact is used to show the approximation ratio of the algorithms for MIN−MAX BCP$_k$ proposed here. When convenient, we denote by $\mathrm{OPT}_k(I)$ the value of an optimal solution for an instance I of MIN−MAX BCP$_k$.

Fact 1 *Any optimal solution for an instance $I = (G, w)$ of* MIN−MAX BCP$_k$ *has value at least $w(G)/k$, that is,* $\mathrm{OPT}_k(I) \geq w(G)/k$.

For $k \geq 3$, let $\mathcal{G}_k$ be the class of connected graphs G containing a cut-vertex v such that $G - v$ has at least $k - 1$ components. We denote by $c(H)$ the number of components of a graph H. The next lemma provides a lower bound for the value of an optimal solution of MIN−MAX BCP$_k$ on instances (G, w) with $G \in \mathcal{G}_k$.

Lemma 1 *Let $I = (G, w)$ be an instance of* MIN−MAX BCP$_k$ *in which $G \in \mathcal{G}_k$, $k \geq 3$, and v is a cut-vertex of G such that $c(G - v) = \ell \geq k - 1$. Let $\mathcal{C} = \{C_i\}_{i\in[\ell]}$ be the set of the components of $G - v$. Suppose further that $w(C_i) \leq w(C_{i+1})$ for every $i \in [\ell - 1]$.*

Then every connected k-partition $\mathcal{P}$ of G satisfies

$$w^+(\mathcal{P}) \geq w(v) + \sum_{i\in[\ell-k+1]} w(C_i).$$

In particular, $\mathrm{OPT}_k(I) \geq w(v) + \sum_{i\in[\ell-k+1]} w(C_i)$.

Proof Consider a connected k-partition $\mathcal{P}$ of G, and let V^* be the class in $\mathcal{P}$ that contains v. Let $q^* := |\{C \in \mathcal{C} \colon V(C) \subseteq V^*\}|$ and $q := |\{C \in \mathcal{C} \colon V(C) \nsubseteq V^*\}|$.

Note that each class in $\mathcal{P} \setminus \{V^*\}$ is either a component of $G - v$ or it is a set properly contained in a unique component of $G - v$. Hence, $q^* + q = \ell$ and $q \leq k - 1$. Therefore, $q^* = \ell - q \geq \ell - k + 1$. Since $w(C_1) \leq w(C_2) \leq \ldots \leq w(C_\ell)$, we conclude that $w(V^*) \geq w(v) + \sum_{i \in [\ell - k + 1]} w(C_i)$, and therefore, $w^+(\mathcal{P}) \geq w(V^*)$. Clearly, it holds that $\mathrm{OPT}_k(I) \geq w(v) + \sum_{i \in [\ell - k + 1]} w(C_i)$. □

Assumption We assume henceforth that in the instances (G, w) of MIN- MAX BCP$_k$, $k \geq 3$, *all vertices have weight at most* $W/2$, *where* W *is the sum of the weights of the vertices in* G. The reason for this is the fact that when G has a vertex v with $w(v) > W/2$, we can easily find an optimal solution. Indeed, if $c(G - v) < k$ then we can take an arbitrary connected $(k - 1)$-partition $\mathcal{P}'$ of $G - v$. The union of $\mathcal{P}'$ with $\{v\}$ forms an optimal connected k-partition of G. If $c(G - v) \geq k$, then Lemma 1 indicates how to obtain an optimal solution. This assumption implies that any set with weight larger than $W/2$ has at least two vertices.

Our algorithm for MIN−MAX BCP$_3$ is inspired by the algorithm proposed by Chen et al. (2021). We adopt basically the same notation used by these authors to refer to the basic operations which are the core of the algorithm.

The strategy used in the algorithm is to start with an arbitrary connected 3-partition and improve it by applying successively (while it is possible) the operations MERGE and PULL, defined in what follows.

We say that a connected 3-partition $\{V_1, V_2, V_3\}$ of G is *ordered* if $w(V_1) \leq w(V_2) \leq w(V_3)$. The input for PULL and MERGE is an ordered connected 3-partition $\{V_1, V_2, V_3\}$ with $w(V_3) > W/2$. The output is also an ordered connected 3-partition. In this context, we say that an ordered 3-partition $\mathcal{P} = \{V_1, V_2, V_3\}$ is *better* than an ordered 3-partition $\mathcal{Q} = \{X_1, X_2, X_3\}$ if either $w(V_3) < w(X_3)$; or $w(V_3) = w(X_3)$ and $|V_3| < |X_3|$.

We say that two classes V_i and V_j are adjacent if there is an edge in G with one endpoint in V_i and the other in V_j. For any $X \subseteq V$, we denote by $N(X)$ the set of vertices in $V \backslash X$ that are adjacent to a vertex of X.

- MERGE($\mathcal{P}$)
  - *Input*: an ordered connected 3-partition $\mathcal{P} = \{V_1, V_2, V_3\}$ of (G, w).
  - *Pre-conditions*: (a) $w(V_3) > w(G)/2$; (b) V_1 and V_2 are adjacent.
  - *Output*: obtain a connected 3-partition $\{V_1 \cup V_2, V_3', V_3''\}$, where $\{V_3', V_3''\}$ is an arbitrary connected 2-partition of $G[V_3]$ with $0 < w(V_3') \leq w(V_3'')$, and return an ordered partition.

From the pre-conditions we have that $w(V_1) + w(V_2) < w(G)/2 < w(V_3)$. Since $w(V_3'') < w(V_3)$, it follows that MERGE($\mathcal{P}$) constructs a 3-connected-partition that is better than $\mathcal{P}$. Note that a depth-first search suffices to check the pre-conditions. Moreover, a connected 2-partition $\{V_3', V_3''\}$ of $G[V_3]$, as desired, can be easily obtained from any spanning tree of this graph (note that V_3 has at least two vertices with positive weights).

- PULL($\mathcal{P}, U, i$)

- *Input*: an ordered connected 3-partition $\mathcal{P} = \{V_1, V_2, V_3\}$ of (G, w), a nonempty subset U of vertices, and $i \in \{1, 2\}$.
- *Pre-conditions*: (a) $w(V_3) > w(G)/2$; (b) $U \subsetneq V_3$, $G[V_i \cup U]$ and $G[V_3 \setminus U]$ are connected; (c) $w(V_i \cup U) < w(V_3)$.
- *Output*: obtain a connected 3-partition $\{V_j, V_i \cup U, V_3 \setminus U\}$ where $j \in \{1, 2\}\setminus\{i\}$, and return an ordered partition.

Observe that if $w(U) > 0$, then all classes in the returned connected 3-partition have weight strictly smaller than $w(V_3)$. If $w(U) = 0$, then $V_3 \setminus U$ is the heaviest class of the new partition and $|V_3 \setminus U| < |V_3|$. Hence PULL$(\mathcal{P}, U, i)$ always produces a connected 3-partition that is better than $\mathcal{P}$. Moreover, it is only executed when a set U satisfying the pre-conditions is given. Thus, this operation can be executed in $\mathcal{O}(|V|)$ time. For simplicity, we say that such a set U is *pull-admissible* (w.r.t. i).

In what follows we describe an algorithm for MIN- MAX- BCP3 (see Algorithm 2) without going into implementation details. It is based on the MERGE and PULL operations, and a routine called PULLCHECK, that either outputs a pull-admissible set, if it exists, or returns an emptyset (indicating that no pull-admissible set exists in that step). In the end of this section, we discuss an efficient implementation of PULLCHECK that runs in linear time.

Algorithm 1 PULLCHECK

```
Input: An ordered connected 3-partition $\mathcal{P} = \{V_1, V_2, V_3\}$ of $(G, w)$ and $i \in \{1, 2\}$
Output: Either a set $U \subset V_3$ that is pull-admissible w.r.t. $i$ or the emptyset $\emptyset$.
1: procedure PULLCHECK$(G, w, \mathcal{P}, i)$
2:     for $v \in N(V_i) \cap V_3$ do
3:         Let $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ be the components of $G[V_3 - v]$ with $w(C_1) \leq \ldots \leq w(C_\ell)$.
4:         Let $U = \{v\} \cup \bigcup_{j \in [\ell-1]} V(C_j)$.
5:         if $w(V_i) + w(U) < w(V_3)$ then
6:             return $U$
7:     return $\emptyset$
```

Lemma 2 *Algorithm 1 on input (G, w), with an ordered connected 3-partition, either returns a pull-admissible set or returns the empty set when no pull-admissible set exists. Moreover, it can be implemented to run in polynomial time.*

Proof By construction, the set U in line 6 is pull-admissible. Now let $T \subsetneq V_3$ be a set that is pull-admissible w.r.t. i. We may assume that $G[T]$ is connected, otherwise, each of its components is pull-admissible and we can consider any of them. Let $v \in T$ be a vertex adjacent to V_i. Clearly, v will be checked at line 2. Let $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ be the components of $G[V_3 - v]$ with $w(C_1) \leq \ldots \leq w(C_\ell)$. (Note that $\ell \geq 1$, as T is a proper subset of V_3.) Since $G[V_3 \setminus T]$ and $G[T]$ are connected, we conclude that $G[T]$ must contain $\ell - 1$ components of $\mathcal{C}$. Moreover, precisely one of the components in $\mathcal{C}$ is not contained in $G[T]$ (but T may contain part of it). (To see this, consider the block structure of $G[V_3]$ and analyze when $\ell = 1$ and $\ell \geq 2$.) It follows from this observation that the set $U = \{v\} \cup \bigcup_{j \in [\ell-1]} V(C_j)$ is such that $w(U) \leq w(T)$. As T is pull-admissible, it holds that $w(V_i) + w(U) \leq w(V_i) + w(T) < w(V_3)$. Hence, at line 6, the set U, which is pull-admissible, will be returned by Algorithm 1.

Since the connected components of $G[V_3 - v]$ can be computed in time $\mathcal{O}(|V| + |E|)$, Algorithm 1 runs in polynomial time. □

Algorithm 2 MIN- MAX- BCP3

```
    Input: An instance $(G, w)$ of MIN−MAX BCP$_3$
    Output: A connected 3-partition of $G$
    Routines: MERGE, PULL and PULLCHECK.
 1: procedure MIN- MAX- BCP3$(G, w)$
 2:     Let $\mathcal{P} = \{V_1, V_2, V_3\}$ be an ordered conn. 3-partition of $G$; $W = w(G)$
 3:     while $w(V_3) > W/2$ do
 4:         if $V_1$ and $V_2$ are adjacent then
 5:             $\mathcal{P} \leftarrow$ MERGE$(\mathcal{P})$   # $\mathcal{P} = \{V_1, V_2, V_3\}$
 6:         else if PULLCHECK$(\mathcal{P}, i)$ returns a set $U \neq \emptyset$ for $i \in [2]$ then
 7:             $\mathcal{P} \leftarrow$ PULL$(\mathcal{P}, U, i)$   # $\mathcal{P} = \{V_1, V_2, V_3\}$
 8:         else
 9:             break
10:     return $\mathcal{P}$
```

Lemma 3 *Algorithm 2 on input (G, w), where $G = (V, E)$ and w is an integer-valued function, outputs a connected 3-partition of G. Moreover, it can be implemented to run in $\mathcal{O}(w(G)|V||E|)$ time.*

Proof The algorithm starts with an arbitrary connected 3-partition of G and only modifies the current partition when a MERGE or PULL operation is performed. As both operations are performed only when the corresponding pre-conditions are satisfied; and by Lemma 2, PULLCHECK finds a pull-admissible set or certifies correctly that no such set exists, Algorithm 2 is correct.

Each time a MERGE operation is executed, the weight of the heaviest class decreases. When a PULL operation is executed, either the weight of the heaviest class V_3 decreases or the weight of the heaviest class remains unchanged but its cardinality decreases (with respect to the previous partition). Thus, if H is the weight of the heaviest class in a certain step of the algorithm, at most $|V|$ PULL Operations may lead to 3-connected partitions with the same weight H. This implies that at most $w(G)|V|$ calls of MERGE or PULL operations are performed by the algorithm. It is immediate that MERGE and PULL can be executed in $\mathcal{O}(|E|)$ time. By Proposition 1 (page 15) the routine PULLCHECK can be implemented to run in $\mathcal{O}(|E|)$ time. Thus, Algorithm 2 can be implemented to run in $\mathcal{O}(w(G)|V||E|)$. □

It is clear that when Algorithm 2 halts and returns a partition $\mathcal{P}$, one of the two cases occurs: (a) either the loop condition in line 3 failed, and in this case, $\mathcal{P}$ has value $w^+(\mathcal{P}) \leq w(G)/2$, or (b) neither MERGE nor PULL operations could be performed (and $w^+(\mathcal{P}) > w(G)/2$). In what follows, we prove that in case (b) the input graph has a particular "star-like" structure which allows us to conclude that the solution produced by the algorithm is optimal.

Lemma 4 *Let $\mathcal{P} = \{V_1, V_2, V_3\}$ be an ordered connected 3-partition produced by Algorithm 2, and let $G_i = G[V_i]$, for $i \in \{1, 2, 3\}$. If $w(V_3) > w(G)/2$, then the following statements hold:*

1. *$w(V_1) < w(G)/4$, and V_1 and V_2 are not adjacent; and*
2. *there exists $u \in V_3$ such that u is a cut-vertex of G, $\{G_1, G_2\} \subseteq \mathcal{C}$, $w(C) \leq w(V_1) \leq w(V_2)$ for each $C \in \mathcal{C} \setminus \{G_1, G_2\}$, where $\mathcal{C}$ is the set of components of $G - u$. Moreover, if $|\mathcal{C}| = 3$ then $w(u) > w(G)/4$.*

Proof Since $w(V_3) > w(G)/2$, the algorithm terminated after executing line 9. This implies that neither MERGE nor PULL operation can be performed on $\mathcal{P}$. Particularly, it means that V_1 and V_2 are not adjacent. Moreover, $w(V_1) + w(V_2) < w(G)/2$, because $w(V_3) > w(G)/2$. Since $w(V_1) \leq w(V_2)$, it follows that $w(V_1) < w(G)/4$. This proves statement 1.

Since G is connected, and V_1 and V_2 are not adjacent, there exists $uv \in E(G)$ such that $u \in V_3$ and $v \in V_1$. Let $\mathcal{C}'$ be the set of components of $G_3 - u$. Note that $\mathcal{C}' \neq \emptyset$ because $w(u) \leq w(G)/2$, and consider a component $C \in \mathcal{C}'$. Let us define $S = \{u\} \cup (\bigcup_{C' \in \mathcal{C} \setminus \{C\}} C')$. Note that, if u is not a cut-vertex of G_3, then $C = G_3 - u$ is the unique component of $\mathcal{C}'$ and $S = \{u\}$. If u is a cut-vertex of G_3, then $\mathcal{C}'$ contains C and other components of $G_3 - u$ distinct from C; hence $G[S] = G_3 - V(C)$. It is clear that $G[C]$ and $G[S]$ are connected subgraphs of G. Since it is not possible to perform PULL$(\mathcal{P}, S, 1)$, it holds that $w(V_3) = w(S) + w(C) \leq w(S) + w(V_1)$. Therefore, we have that $w(C) \leq w(V_1) \leq w(V_2)$.

Suppose, to the contrary, that V_1 is adjacent to C. Thus, the partition $\{V_1 \cup V(C), V_2, V_3 \setminus V(C)\}$ is a connected 3-partition of G. Since $w(C) + w(V_1) \leq 2w(V_1) < w(G)/2$, Algorithm 2 could perform PULL$(\mathcal{P}, V(C), 1)$, a contradiction. Similarly, V_2 is not adjacent to C, otherwise the algorithm could execute PULL$(\mathcal{P}, V(C), 2)$ because $w(C)+w(V_2) \leq w(V_1)+w(V_2) < w(G)/2 < w(V_3)$. By statement 1, V_1 and V_2 are not adjacent, and thus we have $N(V_1) \cap V_3 = N(V_2) \cap V_3 = \{u\}$. Therefore, u is a cut-vertex of G and $\mathcal{C} = \{G_1, G_2\} \cup \mathcal{C}'$. This concludes the proof of statement 2. □

Theorem 1 *Algorithm 2 is a pseudo-polynomial $\frac{3}{2}$-approximation for* MIN- MAX BCP$_3$ *that runs in $\mathcal{O}(w(G)|V||E|)$ time on an instance $(G = (V, E), w)$.*

Proof Let $\mathcal{P} = \{V_1, V_2, V_3\}$ be an ordered 3-partition of G, returned by the algorithm; and let $G_i = G[V_i]$, for $i = 1, 2, 3$. By Lemma 3, $\mathcal{P}$ is indeed a connected 3-partition of G and it can be computed in time $\mathcal{O}(w(G)|V||E|)$. If $w(V_3) \leq w(G)/2$, then it follows directly from Fact 1 that $w^+(\mathcal{P}) = w(V_3) \leq (3/2)\,\mathrm{OPT}_3(G, w)$.

Suppose now that $w(V_3) > w(G)/2$. This means that Algorithm 2 terminated because neither MERGE nor PULL operation could be performed on $\mathcal{P}$. By Lemma 4(2), there exists $u \in V_3$ such that u is a cut-vertex of G, $\{G_1, G_2\} \subseteq \mathcal{C}$, and $w(C) \leq w(V_1) \leq w(V_2)$ for each $C \in \mathcal{C} \setminus \{G_1, G_2\}$, where $\mathcal{C}$ is the set of components of $G - u$. Thus, $w^+(\mathcal{P}) = w(V_3) = w(u) + \sum_{C \in \mathcal{C} \setminus \{G_1, G_2\}} w(C) \leq \mathrm{OPT}_3(G, w)$, where the last inequality follows from Lemma 1. Therefore, in this case, the partition $\mathcal{P}$ produced by the algorithm is an optimal solution for the instance (G, w) of MIN−MAX BCP$_3$. □

We note that Algorithm 2 applied to instances in which $w(v) = 1$ for each vertex v corresponds to the algorithm designed by Chen et al. (2021) for 1- MIN- MAX BCP$_3$. In this case, it runs in time $\mathcal{O}(|V||E|)$ since pull-admissible sets always have positive weight, at most $|V|$ MERGE or PULL operations are performed, and PULLCHECK takes time $\mathcal{O}(|E|)$.

In what follows, we show how to extend the result obtained for MIN- MAX BCP$_3$ to obtain results for MIN−MAX BCP$_k$, for all $k \geq 4$. For simplicity, we say that a vertex u satisfying condition 2 of Lemma 4 is a *star-center*. Moreover, when u is a star-center, we label the ℓ components of $G - u$ as $\mathcal{C} = \{C_1, C_2, \ldots, C_\ell\}$, where $C_\ell = G[V_2]$, $C_{\ell-1} = G[V_1]$ and $w(C_i) \leq w(C_{i+1})$ for all $i \in [\ell - 1]$.

The next algorithm uses a routine called GETSINGLETONS which receives as input a connected graph $G = (V, E)$, a connected k'-partition $\mathcal{P}$ of G, and an integer $q \geq 0$ such that $k' + q \leq |V|$, then it produces a connected $(k' + q)$-partition of G in time $\mathcal{O}(|V||E|)$ (where q of the classes in the partition are singletons).

Algorithm 3 GETSINGLETONS

```
Input: A connected graph $G = (V, E)$, a connected $k'$-partition $\mathcal{P}$ of $(G, w)$, and an integer $q \geq 0$ such that $k' + q \leq |V|$
Output: A connected $(k' + q)$-partition of $G$
1: procedure GETSINGLETONS($G, w, q, \mathcal{P}$)
2:     while $q > 0$ do
3:         Let $U \in \mathcal{P}$ be such that $|U| \geq 2$ and $u$ be a non cut-vertex of $G[U]$.
4:         $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{U\}) \cup (\{\{u\}\} \cup \{U \setminus \{u\}\})$
5:         $q \leftarrow q - 1$
6:     return $\mathcal{P}$
```

Algorithm 4 MIN- MAX- BCPk (fixed k, $k \geq 3$)

```
Input: An instance $(G = (V, E), w)$ of MIN−MAX BCP$_k$
Output: A connected $k$-partition of $G$
Routines: MIN- MAX- BCP3, GETSINGLETONS
1: procedure MIN- MAX- BCP$k(G, w)$
2:     $\mathcal{P} \leftarrow$ MIN- MAX- BCP3$(G, w)$   # $\mathcal{P} = \{V_1, V_2, V_3\}$
3:     if $w^+(\mathcal{P}) \leq w(G)/2$ or $|V_3| = 1$ then
4:         $\mathcal{P}' \leftarrow$ GETSINGLETONS$(G, w, k - 3, \mathcal{P})$
5:     else
6:         Let $u$ be the star-center and let $\mathcal{C} = \{C_i\}_{i \in [\ell]}$ be the components of $G - u$.
7:         if $\ell \geq k - 1$ then
8:             Let $t = \ell - k + 1$ and $V' = (\bigcup_{i \in [t]} V(C_i)) \cup \{u\}$.
9:             $\mathcal{P}' \leftarrow \{V', V(C_{t+1}), \ldots, V(C_{\ell-1}), V(C_\ell)\}$
10:        else
11:            $\mathcal{P} \leftarrow \{\{u\}\} \cup \{C_i\}_{i \in [\ell]}$
12:            $\mathcal{P}' \leftarrow$ GETSINGLETONS$(G, w, k - 1 - \ell, \mathcal{P})$
13:    return $\mathcal{P}'$
```

Theorem 2 *For each integer $k \geq 3$, Algorithm 4 is a pseudo-polynomial $\frac{k}{2}$-approximation for the problem* MIN−MAX BCP$_k$ *that runs in $\mathcal{O}(w(G)|V||E|)$ time on an instance $(G = (V, E), w)$.*

Proof We first note that Algorithm 3 is correct, since while $q > 0$ it iteratively removes q non cut-vertices from nontrivial classes of $\mathcal{P}$ and create singleton classes. As $|V| \geq$

$k' + q$, when $q > 0$ there always exists a nontrivial class $U \in \mathcal{P}$ satisfying the conditions in line 3 (because $\mathcal{P}$ is a k'-connected partition and $k' < |V|$). Note that, since $G[U]$ is connected and nontrivial, we may take a spanning tree in $G[U]$; then, any leaf of such a tree is a non cut-vertex. Thus, Algorithm 3 has time complexity $\mathcal{O}(|V||E|)$.

Now we turn to Algorithm 4. We may assume that $k \geq 4$, since for $k = 3$ the result follows from Theorem 1. Let $\{V_1, V_2, V_3\}$ be the ordered connected 3-partition produced in line 2, and let $G_i = G[V_i]$, for $i = 1, 2, 3$.

If the condition in line 3 is satisfied, then since Algorithm 3 is correct, the partition $\mathcal{P}'$ (in line 4) is a connected k-partition of G. Clearly, if V_3 is a singleton $\{u\}$, then $\mathcal{P}'$ is optimal, since $w(u) \leq \mathrm{OPT}_k(G, w)$. Moreover, when $w^+(\mathcal{P}) \leq w(G)/2$, it holds that $w^+(\mathcal{P}') \leq w(G)/2 \leq (k/2)\,\mathrm{OPT}_k(G, w)$, where the last inequality is justified by Fact 1.

Suppose now that $w(V_3) > w(G)/2$ and $|V_3| \geq 2$. By Lemma 4(2), there exists a star-center $u \in V_3$. Let $\mathcal{C} = \{C_i\}_{i \in [\ell]}$ be the components of $G - u$. We now consider two cases according to the values of k and ℓ.

If $\ell \geq k - 1$, then in any connected k-partition of G the class containing u must also contain $t = \ell - k + 1$ components of $G - u$. The class V' defined in line 8 consists of the union of u and the t lightest such components. Clearly, $\mathcal{P}' := \{V', V(C_{t+1}), \ldots, V(C_{\ell-1}), V(C_\ell)\}$ is a connected k-partition of G, and $w^+(\mathcal{P}') = \max\{w(V'), w(V_2)\}$ (recall that $C_\ell = G[V_2]$). If $w^+(\mathcal{P}') = w(V')$, it follows from Lemma 1 that $\mathcal{P}'$ is an optimal connected k-partition of G. Otherwise, $w^+(\mathcal{P}') = w(V_2) \leq w(G)/2 \leq (k/2)\,\mathrm{OPT}_k(G, w)$.

If $\ell \leq k - 2$, starting with the connected partition $\mathcal{P} = \{\{u\}\} \cup \{C_i\}_{i \in [\ell]}$ (as defined in line 11), using Algorithm 3 we obtain a connected k-partition $\mathcal{P}'$ of G. Clearly, $w^+(\mathcal{P}') \leq w(V_2) \leq w(G)/2$, and so $w^+(\mathcal{P}') \leq (k/2)\,\mathrm{OPT}_k(G, w)$.

Finally, let us analyze the time complexity of Algorithm 4. As we have mentioned, Algorithm 3 has time complexity $\mathcal{O}(|V||E|)$. Since Algorithm 2 (in line 2) is a pseudo-polynomial algorithm for MIN- MAX BCP$_3$ that runs in $\mathcal{O}(w(G)|V||E|)$ time (cf. Theorem 1), we conclude that Algorithm 4 is a pseudo-polynomial $\frac{k}{2}$-approximation for MIN- MAX BCP$_k$ with the same running time. □

By Theorem 2, Algorithm 4 is a (polynomial) $\frac{k}{2}$-approximation if the weights assigned to the vertices are bounded by a polynomial on the order of the graph. In case the weights assigned to the vertices are arbitrary, it is possible to apply a scaling technique and use the previous algorithm as a subroutine to obtain a polynomial algorithm for MIN−MAX BCP$_k$ with approximation ratio $(\frac{k}{2} + \varepsilon)$, for any fixed $\varepsilon > 0$.

We now prove a more general result, applicable to any pseudo-polynomial α-approximation algorithm for MIN−MAX BCP$_k$ whose running time depends on the value of the weights.

Theorem 3 *Let $k \geq 3$ be an integer, and let $I = (G = (V, E), w)$ be an instance of* MIN−MAX BCP$_k$. *If there is a pseudo-polynomial α-approximation algorithm $\mathcal{A}$ for* MIN−MAX BCP$_k$ *that runs in $\mathcal{O}(w(G)^c|V||E|)$ time for some constant c, then Algorithm 5 is an $\alpha(1 + \varepsilon)$-approximation for* MIN−MAX BCP$_k$ *that runs in $\mathcal{O}(|V|^{2c+1}|E|/\varepsilon^c)$ time.*

Algorithm 5 ε-MIN- MAX- BCPk (fixed k, $k \geq 3$)

```
    Input: An instance $(G = (V, E), w)$ of MIN−MAX BCP$_k$
    Output: A connected $k$-partition of $G$
    Routine: a pseudo-polyn. $\alpha$-approx. algorithm $\mathcal{A}$ for MIN−MAX BCP$_k$
1: procedure $\varepsilon$- MIN- MAX- BCP$k(G, w)$
2:     $\theta \leftarrow \max_{v \in V} w(v)$
3:     $\lambda \leftarrow \frac{\varepsilon\theta}{|V|}$
4:     for $v \in V$ do
5:         $\widehat{w}(v) \leftarrow \left\lceil \frac{w(v)}{\lambda} \right\rceil$
6:     $\mathcal{P} \leftarrow \mathcal{A}(G, \widehat{w})$
7:     return $\mathcal{P}$
```

Proof Let $\mathcal{P}^*$ (resp. $\mathcal{P}$) be an optimal solution (resp. a solution produced by Algorithm 5) on input I. Denote by V_k^* and V_k the heaviest classes in $\mathcal{P}^*$ and $\mathcal{P}$, respectively.

First, note that $\mathcal{P}^*$ is a feasible solution for the instance $(G, \widehat{w})$, and so $\widehat{w}^+(\mathcal{P}^*) = \sum_{v \in V_k^*} \widehat{w}(v) \geq \mathrm{OPT}_k(G, \widehat{w})$. Moreover, $\widehat{w}^+(\mathcal{P}) = \sum_{v \in V_k} \widehat{w}(v) \leq \alpha\, \mathrm{OPT}_k(G, \widehat{w})$ since $\mathcal{A}$ is an α-approximation. It is clear from line 5 that $w(v)/\lambda \leq \widehat{w}(v) \leq w(v)/\lambda + 1$ for every $v \in V$. Hence, the following sequence of inequalities hold:

$$\begin{aligned} w^+(\mathcal{P}) &= \sum_{v \in V_k} w(v) \leq \lambda \sum_{v \in V_k} \widehat{w}(v) \leq \lambda\alpha\, \mathrm{OPT}_k(G, \widehat{w}) \\ &\leq \lambda\alpha \sum_{v \in V_k^*} \widehat{w}(v) \leq \lambda\alpha \sum_{v \in V_k^*} \left(\frac{w(v)}{\lambda} + 1\right) \\ &= \alpha\, \mathrm{OPT}_k(G, w) + \lambda\alpha|V_k^*|. \end{aligned} \tag{1}$$

Since $\lambda = \varepsilon\theta/|V|$ (see line 3) and $\theta \leq \mathrm{OPT}_k(G, w)$, it follows from inequality (1) that

$$w^+(\mathcal{P}) \leq \alpha\, \mathrm{OPT}_k(G, w) + \alpha\varepsilon\theta \leq \alpha(1 + \varepsilon)\, \mathrm{OPT}_k(G, w).$$

The running-time of Algorithm 5 is clearly dominated by the running-time of $\mathcal{A}$ on input $(G, \widehat{w})$ in line 6 which takes time $\mathcal{O}(\widehat{w}(G)^c|V||E|)$. It follows from the scaling in line 5 that $\widehat{w}(v) \leq w(v)/\lambda + 1 \leq |V|/\varepsilon + 1$ for every $v \in V$. Therefore, $\widehat{w}(G) \leq |V|^2/\varepsilon + |V|$, and thus, the algorithm runs in $\mathcal{O}(|V|^{2c+1}|E|/\varepsilon^c)$ time. □

Corollary 1 *For each integer $k \geq 3$ and $\varepsilon' > 0$, there is a $(\frac{k}{2} + \varepsilon')$-approximation for* MIN−MAX BCP$_k$ *that runs in $\mathcal{O}(|V|^3|E|/\varepsilon')$ time on an instance $(G = (V, E), w)$.*

Proof The result follows from Theorem 3, by taking Algorithm 5 with $\varepsilon = \varepsilon'/(k/2)$ and Algorithm 4 as the routine $\mathcal{A}$ it requires. The approximation ratio $k/2$ of Algorithm 4 is guaranteed by Theorem 2. □

An algorithm analogous to Algorithm 5 can be designed for MAX−MIN BCP$_k$. In this case, change line 2 to $\theta \leftarrow \min_{v \in V} w(v)$, change line 5 to $\widehat{w}(v) \leftarrow \left\lfloor \frac{w(v)}{\lambda} \right\rfloor$, and

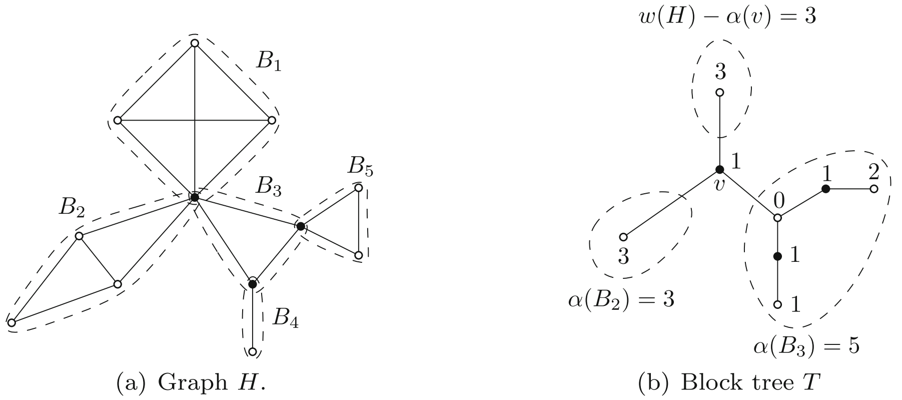

Fig. 1 Illustration of the GetHeaviestComponents algorithm. All vertices of the graph H depicted in (a) have unit weight; those with black filling are the cut-vertices; and the dashed lines indicate the blocks of H. In (b), we show the block tree T of H, and the weights (w_T) of its vertices. For the vertex v shown in T, we consider the tree $\widehat{T}$ rooted at v and indicate by the dashed lines the subtrees of $\widehat{T}$ rooted at each neighbor of v

consider a routine that is a pseudo-polynomial α-approximation for MAX−MIN BCP$_k$. Then, a theorem similar to Theorem 3 can be obtained for MAX−MIN BCP$_k$.

2.1 Finding a pull-admissible set efficiently

In Lemma 3 we mentioned that Algorithm 2 can be implemented to run in $\mathcal{O}(w(G)|V||E|)$ time. To guarantee this result, we show that Algorithm PullCheck can be implemented to run in linear time. We discuss this in what follows.

Note that, the loop starting at line 2 of PullCheck refers to vertices $v \in N(V_i) \cap V_3$. For each such vertex v, there are two possibilities. If v is not a cut-vertex of $G[V_3]$, then the set $U = \{v\}$ is a candidate to be a pull-admissible set, and this can be easily checked. If v is a cut-vertex of $G[V_3]$, then the set U that is the union of $\{v\}$ and the vertex sets of all components of $G[V_3] - v$ except the heaviest one is a candidate for a pull-admissible set. Thus, finding efficiently the heaviest component of $G[V_3] - v$ (when v is a cut-vertex of $G[V_3]$) is a crucial step. Our strategy is to take care of all cut-vertices of $G[V_3]$ at once, by running first a procedure called GetHeaviestComponents.

This procedure is described in what follows for an arbitrary connected graph H. It constructs a block tree T of H, and assigns first a weight $w_T(x)$ to each vertex x of T (see lines 4–5 of Algorithm 6). The tree T has two classes of vertices: the class corresponding to the cut-vertices of H, denoted by $V_C(T)$, and the class corresponding to the nontrivial blocks (biconnected components) of H.

When $V_C(T) \neq \emptyset$, then we consider $\widehat{T}$, the tree T rooted at an arbitrary vertex in $V_C(T)$, and assign to each vertex x in $\widehat{T}$ a value $\alpha(x)$ defined as the weight (w.r.t. w_T) of the subtree of $\widehat{T}$ rooted at x. The function $\alpha(\cdot)$ is used to define a value $h(v)$, for each vertex $v \in V_C(T)$, that corresponds the weight of the heaviest component of $H - v$ (see lines 13–14 of Algorithm 6). The procedure GetHeaviestComponents outputs a pair (T, h) (Fig. 1).

Algorithm 6 GETHEAVIESTCOMPONENTS

```
    Input: A connected graph $H$ and weights $w : V(H) \to \mathbb{Q}_{\geq}$
    Output: A block tree $T$ of $H$ and a function $h : V_C(T) \to \mathbb{Q}$, such that
            $h(v) =$ weight of the heaviest component of $H - v$.
1: procedure GETHEAVIESTCOMPONENTS($H, w$)
2:     Construct a block tree $T$ of $H$
3:     Let $V_C(T)$ be the vertices of $T$ that are cut-vertices of $H$
4:     Compute $w_T : V(T) \to \mathbb{Q}$. If $x \in V_C(T)$ then $w_T(x) := w(x)$; otherwise,

5: $x$ corresponds to a block $B$ of $H$ and $w_T(x) := \sum_{u \in V(B) \setminus V_C(T)} w(u)$
6:     if $V_C(T) = \emptyset$ then
7:         return $(T, \emptyset)$
8:     else
9:         Let $\widehat{T}$ be the tree $T$ rooted at an arbitrary vertex in $V_C(T)$
10:         for $x \in V(\widehat{T})$ do
11:            $\alpha(x) \leftarrow$ weight (w.r.t. $w_T$) of the subtree of $\widehat{T}$ rooted at $x$
12:         for $v \in V_C(T)$ do
13:            Let $u$ be the parent of $v$ in $(\widehat{T})$
14:            $h(v) \leftarrow \max \left\{ \max_{z \in N_T(v) \setminus \{u\}} \{\alpha(z)\}, w(H) - \alpha(v) \right\}$
15:         return $(T, h)$
```

The next algorithm, called PULLCHECK+, gives more detail about the implementation of algorithm PULLCHECK. It uses the algorithm GETHEAVIESTCOMPONENTS described previously.

Algorithm 7 PULLCHECK+

```
    Input: An ordered conn. 3-part. $\mathcal{P} = \{V_1, V_2, V_3\}$ of $(G, w)$, and $i \in \{1, 2\}$.
    Output: Either a set $U \subsetneq V_3$ that is pull-admissible w.r.t. $i$, or the emptyset $\emptyset$.
1: procedure PULLCHECK+($\mathcal{P}, i$)
2:     $(T, h) \leftarrow$ GETHEAVIESTCOMPONENTS($G[V_3], w$)
3:     for $v \in N(V_i) \cap V_3$ do
4:         if $v \in V_C(T)$ and $w(V_i) < h(v)$ then
5:             Let $\{C_1, \ldots, C_\ell\}$ be the components of $G[V_3] - v$ with $w(C_1) \leq \ldots \leq w(C_\ell)$.
6:             return $U := \{v\} \cup \bigcup_{j \in [\ell-1]} V(C_j)$
7:         else if $v \notin V_C(T)$ and $w(V_i) + w(v) < w(V_3)$ then
8:             return $U := \{v\}$
9:     return $\emptyset$
```

It should be noted that the test done in line 4 of PULLCHECK+, if satisfied, indicates that the set U defined in line 6 is indeed a pull-admissible set. Note that for the components defined in line 5, we have that $h(v) = w(V_3) - (w(v) + \sum_{j \in [l-1]} w(C_j))$. Thus, $w(V_i) < h(v)$ is equivalent to $w(V_i) + w(U) < w(V_3)$, where $U = \{v\} \cup \bigcup_{j \in [\ell-1]} V(C_j)$.

Proposition 1 *Algorithm 7 (*PULLCHECK+*) describes an implementation of* PULLCHECK *that receives a connected 3-partition of $(G = (V, E), w)$ and in $\mathcal{O}(|E|)$ time returns either a pull-admissible set or the empty set when no pull-admissible set exists.*

Proof It suffices to note that a block tree of a connected graph $G = (V, E)$ can be constructed in $\mathcal{O}(|E|)$ time, and therefore the routine GETHEAVIESTCOMPONENTS also takes $\mathcal{O}(|E|)$ time. Once (T, h) is constructed, the for-loop of PULLCHECK+ takes time at most $\mathcal{O}(|E|)$. The correctness of PULLCHECK+ is guaranteed by the correctness of PULLCHECK (proved in Lemma 2). □

3 Fractional bipartition and its consequences for MIN−MAX (MAX−MIN) BCP$_2$

In this section we present a simple algorithm, called FRACTIONAL- BIP, that finds (if existent) a desired fractional connected bipartion of a weighted graph (G, w). More precisely, given a fraction p/q such that $2/3 \leq p/q < 1$, the algorithm either returns a connected bipartition $\{V_1, V_2\}$ of G with $w(V_1) \leq w(V_2) \leq \frac{p}{q}W$ or reveals that G has a cut-vertex u and provides information on the weights of the components of $G - u$. Of course in this case, we do not have the notion of balance of the two classes. However, we shall see that such fractional bipartitions lead to a unified approach to design simpler approximation algorithms for both MIN−MAX and MAX−MIN versions of the problem.

As the algorithm for MIN- MAX BCP$_3$, given in the previous section, FRACTIONAL-BIP also iteratively applies an operation to reduce the weight of the heaviest class. This operation is called FRAC- PULL, in analogy to the operation PULL, seen previously. We give its description, and then use it in the Algorithm FRACTIONAL- BIP.

- FRAC- PULL($\mathcal{P}, U, p/q$)
  - Input: an ordered connected bipartition $\mathcal{P} = \{V_1, V_2\}$ of (G, w), $W = w(G)$, a nonempty set U of vertices, and a fraction $p/q \geq 2/3$.
  - Pre-conditions: (a) $w(V_2) > (p/q)W$; (b) $U \subsetneq V_2$, $G[V_1 \cup U]$ and $G[V_2 \setminus U]$ are connected; (c) $w(V_1 \cup U) < w(V_2)$.
  - Output: Set $\{V_1, V_2\} \leftarrow \{V_1 \cup U, V_2 \backslash U\}$ and reorder the classes if necessary to return an ordered partition.

In the next algorithm we refer to a routine called FRAC- PULLCHECK, that is similar to the routine PULLCHECK (Algorithm 1), and therefore will not be described here. It returns (if existent) a nonempty set U that is required in the operation FRAC- PULL. This set is used to update the current bipartition. Then either the algorithm halts with a bipartition $\{V_1, V_2\}$ with $w(V_1) \leq w(V_2) \leq (p/q)W$, or informs that G has a cut-vertex u that provides information on the structure of G. The proof of Lemma 5 indicates how such a cut-vertex can be found, and Proposition 2 guarantees that for $p/q = 2/3$ the bipartition is optimal for both MIN−MAX BCP$_2$ and MAX−MIN BCP$_2$.

Lemma 5 *Algorithm* FRACTIONAL- BIP *on the input* $(G, w, p/q)$*, with* $p/q \geq 2/3$ *and* $W = w(G)$*, produces a connected bipartition* $\mathcal{P} = \{V_1, V_2\}$ *of* G*, for which one of the following statements holds:*

1. $w(V_1) \leq w(V_2) \leq (p/q)W$*; or*
2. $w(V_2) > (p/q)W$*, there exists* $u \in V_2$ *such that* u *is a cut-vertex of* G*, and every connected component* C *of* $G - u$ *is such that* $w(C) \leq w(V_1)$.

Algorithm 8 FRACTIONAL- BIP

```
    Input: A connected graph $(G, w)$ and a fraction $p/q$ such that $2/3 \leq p/q < 1$
    Output: An ordered connected bipartition $\{V_1, V_2\}$ of $G$
    Routines: FRAC- PULL and FRAC- PULLCHECK
1: procedure FRACTIONAL- BIP($G, w, p/q$)
2:     Let $\mathcal{P} = \{V_1, V_2\}$ be an ordered connected bipartition of $G = (V, E)$; $W = w(G)$
3:     while $w(V_2) > (p/q)\, W$ do
4:         if FRAC- PULLCHECK($\mathcal{P}, p/q$) returns a nonempty set $U$ then
5:             $\mathcal{P} \leftarrow$ FRAC- PULL($\mathcal{P}, U, p/q$)   # $\mathcal{P} = \{V_1, V_2\}$
6:         else
7:             return $\mathcal{P}$ and a cut-vertex $u$ of $G$
8:     return $\mathcal{P}$
```

Proof Note that, in the beginning, $\mathcal{P} = \{V_1, V_2\}$ is a connected bipartition, and FRAC- PULL always returns a connected bipartition. If the bipartition $\mathcal{P}$ is returned (at line 8) because the while condition (line 3) is not satisfied, it is immediate that case 1 occurs.

If the bipartition $\mathcal{P}$ is returned within the while loop (line 7), then $w(V_2) > (p/q)W$. Since G is connected, there exists $vu \in E$ such that $v \in V_1$ and $u \in V_2$. Let $\mathcal{C}$ be the set of connected components of $G[V_2 \backslash u]$. Fix $C \in \mathcal{C}$ and let $S = V_2 \backslash C$. (If $|\mathcal{C}| = 1$, then $S = \{u\}$.) Since FRAC- PULL($\{V_1, V_2\}, S, p/q$) could not be performed in the step that preceded the break, we have that $w(V_1) + w(S) \geq w(V_2) = w(C) + w(S)$, and hence

$$w(C) \leq w(V_1).$$

It remains to prove that u is a cut-vertex of G (that is, there is no vertex in V_1 that is adjacent to $V_2 \setminus \{u\}$). For that, suppose that V_1 is adjacent to a component $C \in \mathcal{C}$. Since $w(V_2) > (p/q)W$, we have that $w(V_1) < (1 - p/q)W$. Using the inequality above and the fact that $p/q \geq 2/3$, we obtain that

$$w(V_1) + w(C) \leq 2w(V_1) < 2(1 - p/q)W \leq (p/q)W < w(V_2).$$

But, if $w(V_1) + w(C) < w(V_2)$, then for $U = C$, the pre-conditions of FRAC- PULL operation are satisfied, and the algorithm could have performed FRAC- PULL($\{V_1, V_2\}, C, p/q$), a contradiction. Thus, we conclude that u is a cut-vertex of G (and also of $G[V_2]$). In this case, G has a star-like structure, where u is the "center" and $G - u$ has the connected components V_1 and those in $\mathcal{C}$. □

It is immediate that if the input graph is 2-connected, then the case (2) mentioned in Lemma 5 does not occur. In this case, we obtain the following result.

Corollary 2 *Let $(G, w, p/q)$ be an input to Algorithm* FRACTIONAL- BIP. *If G is 2-connected, $W = w(G)$, $2/3 \leq p/q < 1$, then this algorithm produces a connected bipartition $\{V_1, V_2\}$ of G with $w(V_2) \leq (p/q)W$.*

We recall that $\mathrm{OPT}_2(I)$ denotes the value of an optimal solution for an instance I of MIN−MAX BCP$_2$. In what follows, we denote by $\widehat{\mathrm{OPT}_2}(I)$ the value of an optimal solution for an instance I of MAX−MIN BCP$_2$.

Proposition 2 *Algorithm* Fractional- Bip *used with fraction $p/q = 2/3$ is a pseudo-polynomial algorithm that has approximation ratio $4/3$ for* min- max BCP$_2$*, and approximation ratio $3/2$ for* max−min BCP$_2$.

Proof Let $I = (G, w)$ be an instance of min−max BCP$_2$ or max−min BCP$_2$, $W = w(G)$, and let $\mathcal{P} = \{V_1, V_2\}$ be a bipartition of G returned by Fractional-Bip$(G, w, 2/3)$. By Lemma 5, either case (1) or (2) occurs. If (1) occurs, then $w(V_1) \leq w(V_2) \leq \frac{2}{3}W$.

Case (a) Consider the problem min−max BCP$_2$.

If case (1) occurs, $w^+(\mathcal{P}) = w(V_2) \leq \frac{2}{3}W \leq \frac{2}{3}(2\,\mathrm{OPT}_2(I)) = \frac{4}{3}\,\mathrm{OPT}_2(I)$.
If (1) occurs, then $w(V_2) > (2/3)W$ and V_2 contains a cut-vertex u of G. Moreover, if $\mathcal{C}$ is the set of the connected components of $G[V_2 \setminus \{u\}]$ then $w(C) \leq w(V_1)$ for all $C \in \mathcal{C}$. Considering the weights of the components, and the structure of G, it is easy to see that $\{V_1, V_2\}$ is an optimal solution of min−max BCP$_2$ (it is the unique optimal solution if all components $C \in \mathcal{C}$ are such that $w(C) < w(V_1)$; otherwise, if there exists $C' \in \mathcal{C}$ such that $w(C') = w(V_1)$ then exchanging C' with V_1 in the previous solution, we obtain another optimal solution.)

Case (b) Consider the problem max−min BCP$_2$.

If case (1) occurs, $w^-(\mathcal{P}) = w(V_1) \geq \frac{1}{3}W \geq \frac{1}{3}(2\widehat{\mathrm{OPT}_2}(I)) = \frac{2}{3}\widehat{\mathrm{OPT}_2}(I)$.
If case (2) occurs, analogously to Case (a), we conclude that $\{V_1, V_2\}$ is an optimal solution of max−min BCP$_2$ (that may not be unique, as in Case (a)). /

□

Frac- PullCheck routine can be implemented (as PullCheck routine) to run in linear time. In fact, its implementation is simpler, because this time only two sets V_1 and V_2 are considered. Algorithm Fractional- Bip is a pseudo-polynomial time algorithm as its running time depends on the weights of the vertices. But in the special case in which the weights are uniform (which can be considered one), the algorithm is polynomial.

We observe that the algorithm designed by Chlebíková for max−min BCP$_2$ is essentially for 2-connected graphs. When the input graph is not 2-connected, one has to run the algorithm for each 2-connected component, with new weights on its vertices; then, choose the best solution (translated to the original graph). Algorithm Fractional- Bip does not require that the input graph be 2-connected. Thus, it may be an alternative, specially for the unweighted case, as it will be polynomial, and can be applied directly to any connected graph. In this case, the approximation ratio (for max−min BCP$_2$) changes from 4/3 to 3/2. An analogous observation holds with respect to the algorithm for min−max BCP$_2$ (the ratio changes from 5/4 to 4/3).

4 Parameterized max−min BCP

In this section we design an integer linear program based fixed-parameter tractable (FPT) algorithm for 1- max- min- BCP parameterized by the vertex cover. In this problem, we are given an unweighted graph G, a positive integer k, and a vertex cover X of G. The objective is to find a connected k-partition of G that maximizes the size of the smallest class. We will show that we can formulate this problem as an integer linear program that runs in time doubly exponential in the size of a vertex cover of the input graph.

Let us consider a fixed instance (G, k) of 1- MAX- MIN- BCP and a vertex cover X of G. Let us denote by I the stable set $V(G) \setminus X$. Recall that we assume that $k \leq |V(G)| = |X| + |I|$. From now on, we assume that $k \leq |X|$ and $|X| \geq 2$. The reason for this is that if $k > |X|$, then there are at least $k - |X|$ classes of size exactly 1 contained in I, and so an optimal solution (which has value equal to 1) can be easily computed; and if $|X| = 1$, then G is a star, and so it is trivial to compute an optimal solution.

Before presenting the details of the proposed algorithm, we prove a simple lemma that guarantees the existence of an optimal solution in which each class intersects the given vertex cover X.

Lemma 6 *Let (G, k) be an instance of* 1- MAX- MIN- BCP *and let X be a vertex cover of G. Then, there exists an optimal connected k-partition $\{V_i\}_{i\in[k]}$ of G such that $V_i \cap X \neq \emptyset$ for all $i \in [k]$.*

Proof Suppose to the contrary that no such partition exists. Let $\{V_i'\}_{i\in[k]}$ be a connected k-partition of G with the smallest number of classes contained in I, and let $V_j' = \{v\} \subseteq I$, for some $j \in [k]$, be one of these classes.

Since $k \leq |X|$, there exists $\ell \in [k] \setminus \{j\}$ such that $|V_\ell' \cap X| \geq 2$. One may easily find a partition $\{V_{\ell,1}', V_{\ell,2}'\}$ of V_ℓ' such that, for $i \in \{1, 2\}$, $G[V_{\ell,i}']$ is connected and $V_{\ell,i}' \cap X \neq \emptyset$. If $N(v) \cap V_\ell' \neq \emptyset$, then assume without loss of generality that $N(v) \cap V_{\ell,1}' \neq \emptyset$. In this case, there is a connected k-partition $\{V_i\}_{i\in[k]}$ of G such that $V_j = V_{\ell,1}' \cup \{v\}$, $V_\ell = V_{\ell,2}'$ and $V_i = V_i'$ for every $i \in [k]\setminus\{j, \ell\}$. If $N(v) \cap V_\ell' = \emptyset$, then there exists $t \in [k]\setminus\{j, \ell\}$ such that $N(v) \cap V_t' \neq \emptyset$ since G is connected. Clearly, such a class intersects X, that is, $V_t' \cap X \neq \emptyset$. Thus, there is a connected k-partition $\{V_i\}_{i\in[k]}$ of G such that $V_j = V_{\ell,1}'$, $V_\ell = V_{\ell,2}'$, $V_t = V_t' \cup \{v\}$ and $V_i = V_i'$ for every $i \in [k]\setminus\{j, \ell, t\}$. In both cases, the partition has a smaller number of classes contained in I than $\{V_i'\}_{i\in[k]}$, a contradiction to the choice of this partition. □

We next use hypergraphs to model the constraints of our formulation for 1- MAX- MIN- BCP. A hyperpath of length m between two vertices u and v in a hypergraph H is a set of hyperedges $\{e_1, \ldots, e_m\} \subseteq E(H)$ such that $u \in e_1$, $v \in e_m$, and $e_i \cap e_{i+1} \neq \emptyset$ for each $i \in \{1, \ldots, m-1\}$. A set of hyperedges $F \subseteq E(H)$ is a (u, v)-cut if there is no hyperpath between u and v in $H - F$.

For each $S \subseteq X$, we define $I(S) = \{v \in I : N(v) = S\}$. Let $u, v \in X$ be a pair of non-adjacent vertices in G, and let $\Gamma_X(u, v)$ be the set of all separators of u and v in $G[X]$. Consider a separator $Z \in \Gamma_X(u, v)$, and denote by $\mathcal{C}(Z)$ the set of components of $G[X - Z]$. Now suppose that for some $i \in [k]$, the subgraph $G[V_i]$ contains a $u - v$ path P that avoids Z. Then P contains at least one vertex in I. Loosely speaking, P uses vertices in I as "bridges" between the components in $\mathcal{C}(Z)$. This observation motivates the following construction. Let H_Z denote the hypergraph with vertices $\mathcal{C}(Z)$ such that, for each $S \subseteq X$ with $I(S) \neq \emptyset$, there is a

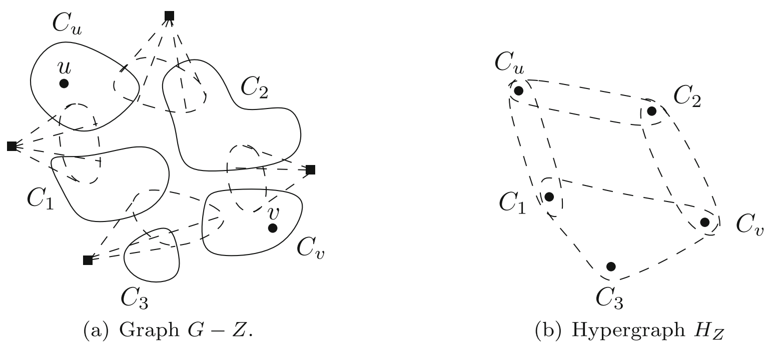

Fig. 2 Illustration of the hypergraph construction. Continuous lines indicate the components in $\mathcal{C}(Z)$. Subsets S of X are represented with dashed lines and their corresponding vertices $I(S)$ are depicted with squares. In this example, the pairs of hyperedges are precisely the minimal (C_u, C_v)-cuts in H_Z

hyperedge $\{C \in \mathcal{C}(Z) \colon S \cap V(C) \neq \emptyset\}$ in H_Z. We denote by $\Lambda_Z(u, v)$ the set of all (C_u, C_v)-cuts in H_Z, where C_u and C_v are the components of $G[X - Z]$ containing u and v, respectively (Fig. 2).

In the proposed formulation, for each $v \in X$ and $i \in [k]$, there is a binary variable $x_{v,i}$ that equals 1 if and only if v belongs to the i-th class of the partition. Moreover, for every $S \subseteq X$ and $i \in [k]$, there is an integer variable $y_{S,i}$ that equals the amount of vertices in $I(S)$ that are assigned to the i-th class. The intuition for the meaning of the y-variables is that all vertices in $I(S)$, for a fixed $S \subseteq X$, play essentially the same role in a connected partition. Hence, the formulation does not need to have decision variables associated with the vertices in $I(S)$, and so it only has an integer variable to count the number of these vertices that are chosen to be in each class. The idea of using integer variables to count indistinguishable vertices in a stable set was used before by Fellows et al. (2008) for the IMBALANCE problem.

Let us define $\eta = 2^{|X|}$, that is, η is the number of subsets of X. Let $\mathcal{B}(G, X, k)$ be the set of vectors in $\mathbb{R}^{(|X|+\eta)k}$ that satisfy the following inequalities (2)–(8).

$$\sum_{v \in X} x_{v,i} + \sum_{S \subseteq X} y_{S,i} \leq \sum_{v \in X} x_{v,i+1} + \sum_{S \subseteq X} y_{S,i+1} \quad \forall i \in [k-1], \tag{2}$$

$$\sum_{i \in [k]} x_{v,i} = 1 \quad \forall v \in X, \tag{3}$$

$$x_{u,i} + x_{v,i} - \sum_{z \in Z} x_{z,i} - \sum_{S \in F} y_{S,i} \leq 1 \quad \forall uv \notin E(G[X]), Z \in \Gamma_X(u, v), F \in \Lambda_Z(u, v), i \in [k], \tag{4}$$

$$y_{S,i} \leq |I(S)| \left(\sum_{v \in S} x_{v,i} \right) \quad \forall S \subseteq X, i \in [k], \tag{5}$$

$$\sum_{i \in [k]} y_{S,i} = |I(S)| \quad \forall S \subseteq X, \tag{6}$$

$$x_{v,i} \in \{0, 1\} \quad \forall v \in X \text{ and } i \in [k], \tag{7}$$

$$y_{S,i} \in \mathbb{Z}_{\geq} \quad \forall S \subseteq X \text{ and } i \in [k]. \tag{8}$$

Inequalities (2) establish a non-decreasing ordering of the classes according to their sizes. Inequalities (3) and (6) guarantee that every vertex of the graph belongs to exactly one class (i.e. the classes define a partition). Due do Lemma 6, we may consider only partitions such that each of its classes intersects X. Thus, whenever a vertex in the stable set I is chosen to belong to some class, at least one of its neighbors in X has to be in the same class. This explains the meaning of inequalities (5). Inequalities (4) guarantee that each class of the partition induces a connected subgraph. This will be clearer in the proof of Lemma 7 below.

Lemma 7 *Let G be a connected graph, let $k \geq 2$ be an integer, and let X be a vertex cover of G. The problem* 1- MAX- MIN- BCP *on instance (G, k) is equivalent to*

$$\max\left\{\sum_{v\in X} x_{v,1} + \sum_{S\subseteq X} y_{S,1} : (x, y) \in \mathcal{B}(G, X, k)\right\}.$$

Proof Let $\{V_i\}_{i\in[k]}$ be a connected k-partition of G such that $V_i \cap X \neq \emptyset$ for every $i \in [k]$. Suppose further that the classes in this partition are ordered so that $|V_i| \leq |V_{i+1}|$ for all $i \in [k-1]$. From $\{V_i\}_{i\in[k]}$, we construct a vector $(\bar{x}, \bar{y}) \in \mathbb{R}^{|X|k} \times \mathbb{R}^{\eta k}$ such that its non-null entries are precisely defined as follows. For each $i \in [k]$, we set $\bar{x}_{v,i} = 1$ for every $v \in X \cap V_i$, and $\bar{y}_{S,i} = |I(S) \cap V_i|$ for every $S \subseteq X$.

We next show that $(\bar{x}, \bar{y})$ satisfies inequalities (2)–(8). It easily follows from the construction of the vector and from the hypothesis on $\{V_i\}_{i\in[k]}$ that inequalities (2),(3),(6),(7), and (8) hold for $(\bar{x}, \bar{y})$. Moreover, since $V_i \cap X \neq \emptyset$ for every $i \in [k]$, inequalities (5) hold for $(\bar{x}, \bar{y})$. It remains to prove that inequalities (4) are satisfied.

Consider an integer $i \in [k]$ such that there is a pair of non-adjacent vertices $u, v \in X \cap V_i$. Let $Z \subseteq X\backslash\{u, v\}$ be a separator of u and v in $G[X]$. Since $G[V_i]$ is connected, there exists a path P in this graph with endpoints u and v such that either $V(P) \cap Z \neq \emptyset$ or $V(P) \cap I \neq \emptyset$. Suppose that $V(P) \cap Z = \emptyset$, otherwise inequalities (4) for Z are clearly satisfied by $(\bar{x}, \bar{y})$. Hence, there is a hyperpath in H_Z linking C_u and C_v, where C_u and C_v are the components of $G[X - Z]$ containing u and v, respectively. As a consequence, for each cut F separating C_u and C_v in the hypergraph H_Z, there exists a vertex $z \in V(P) \cap I$ such that $N(z) \in F$, and so $\bar{y}_{N(z),i} \geq 1$. Therefore, inequalities (4) are satisfied by $(\bar{x}, \bar{y})$.

Let $(\bar{x}, \bar{y}) \in \mathcal{B}(G, X, k)$. Consider a subset $S \subseteq X$. It follows from inequality (6) for S that there is a partition $\{I_i(S)\}_{i\in[k]}$ of $I(S)$ such that $|I_i(S)| = \bar{y}_{S,i}$. We remark that some classes in this partition may be empty. For each $i \in [k]$, let us define $V_i = (\bigcup_{S\subseteq X} I_i(S)) \cup \{v \in X : \bar{x}_{v,i} = 1\}$. One may easily verify that $|V_i| = \sum_{v\in X} \bar{x}_{v,i} + \sum_{S\subseteq X} \bar{y}_{S,i}$. Observe now that $I(S) \cap I(S') = \emptyset$ for all S, S' subsets of X with $S \neq S'$, and thus $\{I(S)\}_{S\subseteq X}$ is a partition of I with possibly some empty classes. Furthermore, inequalities (3) guarantee that each vertex in X belongs to exactly one class V_i, for some $i \in [k]$. Therefore, $\{V_i\}_{i\in[k]}$ is a partition of the vertices in G. Because of inequalities (2), we also have that $|V_i| \leq |V_{i+1}|$ for all $i \in [k-1]$. We shall prove that $G[V_i]$ is connected for each $i \in [k]$.

Suppose, to the contrary, that there exists $i \in [k]$ such that $G[V_i]$ is not connected. Because of inequalities (5), every component of $G[V_i]$ has to intersect X. Let us define $Z = X \setminus V_i$, and let H_Z be the hypergraph of the components of $G[X - Z]$ as defined earlier. It follows that, for each hyperedge S of H_Z, no vertex in $I(S)$ belongs to V_i. Hence, for every pair of vertices $u, v \in V_i \cap X$ belonging to distinct components of $G[V_i]$, it holds that

$$\bar{x}_{u,i} + \bar{x}_{v,i} - \sum_{z \in Z} \bar{x}_{z,i} - \sum_{S \in E(H_Z)} \bar{y}_{S,i} = 2 > 1.$$

This is a contradiction to the fact that $(\bar{x}, \bar{y})$ satisfies inequalities (4). As a consequence, we conclude that $G[V_i]$ is connected for each $i \in [k]$. Therefore $\{V_i\}_{i \in [k]}$ is a connected k-partition of G.

Finally, it follows from Lemma 6 that the proposed integer linear program has an optimal solution that corresponds to an optimal connected k-partition of G. As a consequence, it is equivalent to solving 1- MAX- MIN- BCP on instance (G, k). □

The main tool to design fixed-parameter tractable algorithms using Integer Linear Programing (ILP) is a theorem due to Lenstra (1983) which shows that checking the feasibility of an ILP problem with a fixed number of variables can be solved in polynomial time. The time and space complexity of Lenstra's algorithm were later improved by Kannan (1987), and Frank and Tardos (1987). In this work, we consider the following optimization version of their results.

In the ILP problem, we are given as input a matrix $A \in \mathbb{Z}^{p \times q}$, vectors $b \in \mathbb{Z}^p$ and $c \in \mathbb{Z}^q$. The objective is to find a vector $x \in \mathbb{Z}^q$ that satisfies all inequalities (i.e. $Ax \leq b$), and maximizes $c^T x$. Let us denote by L the size of the binary representation of an input (A, b, c) of the problem.

We next present the maximization version of the theorem showed in Cygan et al. (2015) on the existence of an FPT algorithm for an ILP problem parameterized by the number of variables.

Theorem 4 (Cygan et al. 2015) *An integer linear programming instance of size L with q variables can be solved using $\mathcal{O}(q^{2.5q+o(q)} \cdot (L + \log M_x) \log(M_x M_c))$ arithmetic operations and space polynomial in $L + \log M_x$, where M_x is an upper bound on the absolute value a variable can take in a solution, and M_c is the largest absolute value of a coefficient in the vector c.*

The previous theorem is now used to show that 1- MAX- MIN- BCP admits an algorithm that runs in time doubly exponential in the size of a vertex cover of the input graph.

Theorem 5 *The problem* 1- MAX- MIN- BCP, *parameterized by the size of a vertex cover of the input graph, is fixed-parameter tractable.*

Proof Consider an instance (G, k) of MAX- MIN- BCP, and a vertex cover X of G. From Lemma 7, $\max \left\{\sum_{v \in X} x_{v,1} + \sum_{S \subseteq X} y_{S,1} : (x, y) \in \mathcal{B}(G, X, k)\right\}$ is equivalent to solving instance (G, k). Observe now that the size of this integer linear program is $2^{2^{\mathcal{O}(|X|)}} \log|G|$. By Theorem 4, it can be solved in time $2^{2^{\mathcal{O}(|X|)}} |G|^{\mathcal{O}(1)}$. Therefore 1- MAX- MIN- BCP is fixed parameter-tractable when parameterized by the size of a vertex cover of the input graph. □

We conclude this section mentioning results on parameterized complexity of the related problem called *Equitable Connected Partition* (ECP), obtained by Enciso et al. (2009). In this problem, it is given a graph G and an integer k, and one looks for a connected k-partition of G into classes whose cardinalities differ by at most one. Note that, k is part of the input. These authors proved that ECP is FPT when parameterized by the minimum size of a vertex cover, and also when parameterized by the maximum number of leaves of a spanning tree of G. They also proved that ECP is W[1]-hard when parameterized by the pathwidth of G, or the minimum size of feedback vertex set of G, or some other parameters.

5 Lower bounds for BCP

We now derive lower bounds on the complexity of finding balanced connected partitions. To this end, let us define the decision version of MAX−MIN BCP.

Problem 1 MAX- MIN BCP DECISION

INSTANCE: a tuple (G, w, k, t) consisting of a connected graph $G = (V, E)$, a weight function $w: V \to \mathbb{Q}_{\geq}$, an integer $k \geq 2$ and $t \in \mathbb{Q}$.

QUESTION: is there a connected k-partition $\mathcal{P}$ of G such that $w^-(\mathcal{P}) \geq t$?

We define MIN−MAX BCP- D analogously, changing the question to the existence of a connected k-partition $\mathcal{P}$ such that $w^+(\mathcal{P}) \leq t$. We simply refer to BCP- D, when both maximization and minimization versions are included.

Let Φ denote a Boolean formula in conjunctive normal form (CNF). We say that Φ is a q-CNF formula if each of its clauses contains exactly q literals. We denote by q- SAT the problem of deciding whether a q-CNF formula is satisfiable; and refer to an N-variable q- SAT an instance of q- SAT with N variables. For every integer $q \geq 3$, we define

$$s_q = \inf\{\delta : \text{there exists a } 2^{\delta N}\text{-time algorithm to solve any } N\text{-variable } q\text{- SAT}\}.$$

Conjecture 1 (Exponential-Time Hypothesis (ETH) Impagliazzo and Paturi (2001))

$$s_3 > 0,$$

or equivalently, there exists an $\varepsilon > 0$ such that 3- SAT cannot be solved in time $\mathcal{O}(2^{\varepsilon N})$ on N-variable instances.

Lemma 8 (Linear-time reduction from 3- SAT to BCP- D) *Let Φ be an instance of* 3- SAT *consisting of N variables and M clauses. There exists a $\mathcal{O}(N + M)$-time algorithm that constructs from Φ an instance $(G, w, 2, \tau)$ of* BCP- D *with $|V(G)| = 3N + M + 2$, such that Φ is satisfiable if, and only if, $(G, w, 2, \tau)$ is a* YES-*instance of* BCP- D.

Proof Let $\varPhi = (X, \mathcal{C})$ be an instance of 3- SAT consisting of a set X of N variables and a set $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ of M clauses. Let G be the graph obtained from $(X, \mathcal{C})$ as follows:

- for each variable $x \in X$, the graph G contains vertices v_x and $v_{\bar{x}}$, and a third vertex α_x which is adjacent to both v_x and $v_{\bar{x}}$;
- for each clause $C_i \in \mathcal{C}$, the graph G has a vertex C_i which is linked to a vertex v_z if, and only if, z is a literal appearing in C_i;
- G has two additional vertices T and F such that both are adjacent to v_x and $v_{\bar{x}}$ for every $x \in X$.

Note that G is a bipartite graph (see Fig. 3). Take $\tau = (N+1)(2N+M)$, and define a weight function $w\colon V(G) \to \mathbb{Q}_{\geq}$ as follows: $w(T) := \tau - N - M$, $w(F) := \tau - N - N(2N+M)$, and $w(v_x) = w(v_{\bar{x}}) = w(C_i) := 1$ and $w(\alpha_x) := 2N+M$ for all $x \in X$ and $C_i \in \mathcal{C}$. This reduction produces a graph with $w(G) = 2\tau$, $|V(G)| = 3N+M+2$ and and $|E(G)| = 4N + 3M$. Therefore, the instance $(G, w, 2, \tau)$ of BCP- D can be constructed in time $\mathcal{O}(N + M)$.

Suppose there exists a connected 2-partition $\mathcal{P} = \{V_0, V_1\}$ of G such that $w^-(\mathcal{P}) \geq \tau$ (or $w^+(\mathcal{P}) \leq \tau$). As $w(G) = 2\tau$, we have that $w^-(\mathcal{P}) = w^+(\mathcal{P}) = \tau$. Note first that the vertices T and F do not belong to the same class of $\mathcal{P}$, since T and F are not adjacent in G and $w(T) + w(F) = \tau$. We now assume without loss of generality that $T \in V_1$ and $F \in V_0$. If there was a variable $x \in X$ such that $\alpha_x \notin V_0$, then we would have $w(V_0) \leq 2\tau - w(T) - w(\alpha_x) = 2\tau - (\tau - N - M) - (2N+M) = \tau - N \leq \tau - 1$, a contradiction to the choice of $\mathcal{P}$. Thus, α_x belongs to V_0 for every $x \in X$. This implies that, for each $x \in X$, the class V_1 cannot contain both v_x and $v_{\bar{x}}$ since these vertices separate α_x from F; thus, $|\{v_x, v_{\bar{x}}\} \cap V_1| \leq 1$. Suppose that $\{v_x, v_{\bar{x}}\} \cap V_1 = \emptyset$ for some $x \in X$. In this case, the class V_1 has weight $w(V_1) \leq w(T) + M + N - 1 = \tau - 1$, a contradiction. Hence, V_1 contains either v_x or $v_{\bar{x}}$ for all $x \in X$. Finally, one may easily verify that $C_i \in V_1$ for all $C_i \in \mathcal{C}$.

Let $\rho\colon X \to \{\text{TRUE}, \text{FALSE}\}$ be such that $\rho(x) := \text{TRUE}$ if, and only if, $v_x \in V_1$. From the construction of G, we conclude that $\varPhi$ is satisfiable with the assignment ρ.

Suppose now that there is an assignment $\rho\colon X \to \{\text{TRUE}, \text{FALSE}\}$ that satisfies $(X, \mathcal{C})$. Construct a vertex partition $\mathcal{P} = \{V_0, V_1\}$ of G as follows:

$$\begin{aligned} V_0 &= \{F\} \cup \{v_x : x \in X \text{ and } \rho(x) = \text{FALSE}\} \\ &\quad \cup \{v_{\bar{x}} : x \in X \text{ and } \rho(x) = \text{TRUE}\} \cup \{\alpha_x : x \in X\}, \\ V_1 &= \{T\} \cup \{v_x : x \in X \text{ and } \rho(x) = \text{TRUE}\} \\ &\quad \cup \{v_{\bar{x}} : x \in X \text{ and } \rho(x) = \text{FALSE}\} \cup \{C_i : C_i \in \mathcal{C}\}. \end{aligned}$$

It is clear that V_0 induces a connected subgraph of G, and that $w(V_0) = w(V_1) = \tau$. For each clause $C \in \mathcal{C}$, there is a variable $x \in X$ such that either literal $x \in C$ and $\rho(x) = \text{TRUE}$, or literal $\bar{x} \in C$ and $\rho(x) = \text{FALSE}$. In both cases, there is a path between every clause vertex C_i and T in $G[V_1]$. Therefore, V_1 induces a connected subgraph of G, and thus $\mathcal{P}$ is a connected 2-partition of G. □

Figure 3 illustrates the reduction defined in Lemma 8 and shows a solution to the given 3-CNF formula: $\rho(x) = \rho(z) = \text{TRUE}$ and $\rho(y) = \text{FALSE}$. The blue

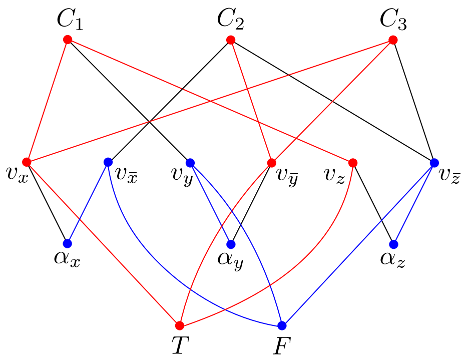

Fig. 3 A solution (in blue and red) to the instance $(G, w, 2, \tau)$ of BCP- D obtained from the 3- SAT instance $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (x \vee \bar{y} \vee \bar{z})$

and red subgraphs, $G[V_0]$ and $G[V_1]$, correspond to the classes of the connected 2-partition $\mathcal{P} = \{V_0, V_1\}$ (without the edges with endpoint T or F that are not in $G[V_0]$ or $G[V_1]$).

Note that the reduction from q- SAT(on N variables and M clauses) to BCP- D in Lemma8 constructs a graph G whose order depends on N and M. Roughly speaking, the next lemma shows that to check whether a q-CNF formula Φ with N variables is satisfiable, one can check whether some q-CNF formulas ϕ_i with N variables and $\mathcal{O}(N)$ clauses are satisfiable. Thus, the reductions corresponding to these formulas ϕ_i construct graphs with order depending only on N, as it will become clear in the proof of Theorem 7.

Theorem 6 (Sparsification Lemma. Impagliazzo et al. (2001)) *Let $q \in \mathbb{N}$, $\gamma > 0$, and Φ be a q-CNF formula with N variables. There is a constant $c = c(q, \gamma)$, and an algorithm $\mathcal{A}$ such that*

1. *$\mathcal{A}$ computes q-CNF formulas $\phi_1, \ldots, \phi_\ell$ from Φ, where $\ell \leq 2^{\gamma N}$;*
2. *$\mathcal{A}$ runs in time $\mathcal{O}^*(2^{\gamma N})$;*
3. *ϕ_i has N variables and at most cN clauses for every $i \in [\ell]$;*
4. *Φ is satisfiable if, and only if, for some $i \in [\ell]$, the formula ϕ_i is satisfiable.*

Theorem 7 *There exists an $\varepsilon' > 0$ such that* MAX−MIN *(resp.* MIN−MAX*)* BCP- D *cannot be solved in time $\mathcal{O}(2^{\varepsilon' n})$ on n-vertex bipartite graphs even when $k = 2$, unless ETH fails.*

Proof Suppose to the contrary that the statement is false. We shall prove that, in this case 3- SAT on N variables can be solved in time $\mathcal{O}(2^{\varepsilon N})$ for all $\varepsilon > 0$.

Consider an arbitrary constant $\varepsilon > 0$, and let Φ be a 3-CNF formula with $N > 2$ variables (an instance of 3- SAT). Take $\gamma = \varepsilon/2$, $c = c(3, \gamma)$ and let $\phi_1, \ldots, \phi_\ell$ with $\ell \leq 2^{\gamma N}$ be the 3-CNF formulas, computed from Φ (in time $\mathcal{O}^*(2^{\gamma N})$), as mentioned in Theorem 6.

For each $i \in [\ell]$, take the formula ϕ_i and construct an instance $(G_i, w_i, 2, t_i)$ of BCP- D using the reduction described in Lemma 8. Since ϕ_i has N variables and at most cN clauses, the graph G_i is such that $|V(G_i)| \leq 3N + cN + 2 \leq (4 + c)N$.

Take $\rho = \varepsilon/2(4 + c)$. By our assumption, there exists an algorithm $\mathcal{B}$ that solves BCP- D in time $\mathcal{O}(2^{\rho n})$, where n is the number of vertices of the input graph.

Hence, the running-time of $\mathcal{B}$ on input $(G_i, w_i, 2, t_i)$ is

$$\mathcal{O}(2^{\rho|V(G_i)|}) = \mathcal{O}(2^{\rho(4+c)N}) = \mathcal{O}(2^{\varepsilon N/2}).$$

To decide whether Φ is satisfiable, we run $\mathcal{B}$ on $(G_i, w_i, 2, t_i)$ for every $i \in [\ell]$. By Theorem 6 and Lemma 8, Φ is satisfiable if, and only if, for some i we have that $(G_i, w_i, 2, t_i)$ is a YES-instance of BCP- D. The total running-time of such algorithm is

$$\mathcal{O}(\ell 2^{\varepsilon N/2}) = \mathcal{O}(2^{\gamma N} 2^{\varepsilon N/2}) = \mathcal{O}(2^{(\gamma+\varepsilon/2)N}) = \mathcal{O}(2^{\varepsilon N}).$$

Thus, the existence of the algorithm $\mathcal{B}$ for BCP- D, as we supposed above, implied that 3- SAT on N variables can be solved in time $\mathcal{O}(2^{\varepsilon N})$ for all $\varepsilon > 0$, a contradiction to ETH. □

6 Concluding remarks

We presented an FPT algorithm for 1- MAX- MIN- BCP parameterized by the size of a vertex cover of the input graph. Its time complexity is doubly exponential in the size of the vertex cover. It would be interesting to see whether other parameters would lead to better complexity for this version and also for the weighted version. The corresponding minimization versions (unweighted and weighted) were not studied under this perspective. It is not clear whether these versions may lead to interesting results.

Acknowledgements The authors would like to thank the referees for the remarks and suggestions that contributed to improve the presentation of this work.

Author Contributions All authors contributed equally to the study conception and design. All authors read and approved the final manuscript.

Funding P.F.S. Moura is supported by Fundação de Amparo à Pesquisa do Estado de Minas Gerais - FAPEMIG (Proc. APQ-01040-21). Y. Wakabayashi is supported by the National Council for Scientific and Technological Development—CNPq (Proc. 423833/2018-9 and 311892/2021-3) and Grant #2015/11937-9, São Paulo Research Foundation (FAPESP).

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing interests The authors have no relevant financial or non-financial interests to disclose.

References

Alimonti P, Calamoneri T (1999) On the complexity of the max balance problem. In: Argentinian workshop on theoretical computer science (WAIT'99), pp 133–138

Becker RI, Perl Y (1983) Shifting algorithms for tree partitioning with general weighting functions. J Algorithms 4(2):101–120

Becker RI, Schach SR, Perl Y (1982) A shifting algorithm for min-max tree partitioning. J ACM 29(1):58–67

Becker RI, Lari I, Lucertini M, Simeone B (1998) Max-min partitioning of grid graphs into connected components. Networks 32(2):115–125

Becker R, Lari I, Lucertini M, Simeone B (2001) A polynomial-time algorithm for max-min partitioning of ladders. Theory Comput Syst 34(4):353–374

Casel K, Friedrich T, Issac D, Niklanovits A, Zeif Z (2021) Balanced crown decomposition for connectivity constraints. In: Mutzel P, Pagh R, Herman G (eds), 29th annual European symposium on algorithms (ESA 2021), volume 204 of Leibniz international proceedings in informatics (LIPIcs), pp 26:1–26:15

Chataigner F, Salgado LRB, Wakabayashi Y (2007) Approximation and inapproximability results on balanced connected partitions of graphs. Discrete Math Theor Comput Sci 9(1):177–192

Chen G, Chen Y, Chen Z-Z, Lin G, Liu T, Zhang A (2020) Approximation algorithms for the maximally balanced connected graph tripartition problem. J Comb Optim 1–21

Chen Y, Chen Z, Lin G, Xu Y, Zhang A (2021) Approximation algorithms for maximally balanced connected graph partition. Algorithmica 83(12):3715–3740

Chlebíková J (1996) Approximating the maximally balanced connected partition problem in graphs. Inf Process Lett 60(5):225–230

Cygan M, Fomin FV, Kowalik Ł, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) Miscellaneous. Springer International Publishing, Cham, pp 129–150

Dyer M, Frieze A (1985) On the complexity of partitioning graphs into connected subgraphs. Discrete Appl Math 10(2):139–153

Enciso R, Fellows MR, Guo J, Kanj IA, Rosamond FA, Suchý O (2009) What makes equitable connected partition easy. In: Chen J, Fomin FV (eds) Parameterized and exact computation, 4th international workshop, IWPEC 2009, Copenhagen, Denmark, September 10–11, 2009, Revised Selected Papers, vol 5917. Lecture notes in computer science. Springer, Berlin, pp 122–133

Fellows MR, Lokshtanov D, Misra N, Rosamond FA, Saurabh S (2008) Graph layout problems parameterized by vertex cover. In: Proceedings of the 19th international symposium on algorithms and computation, ISAAC '08. Springer, Berlin, pp 294–305

Frank A, Tardos É (1987) An application of simultaneous diophantine approximation in combinatorial optimization. Combinatorica 7(1):49–65

Frederickson GN (1991) Optimal algorithms for tree partitioning. In: Proceedings of the second annual ACM-SIAM symposium on discrete algorithms, SODA '91, pp 168–177

Győri E (1978) On division of graph to connected subgraphs. In: Combinatorics (Proc. Fifth Hungarian Colloq., Koszthely, 1976), volume 18 of Colloq. Math. Soc. János Bolyai, pp 485–494

Holyer A (2019) On the independent spanning tree conjectures and related problems. Ph.D. thesis, Georgia Institute of Technology, School of Mathematics

Impagliazzo R, Paturi R (2001) On the complexity of k-SAT. J Comput Syst Sci 62(2):367–375

Impagliazzo R, Paturi R, Zane F (2001) Which problems have strongly exponential complexity? J Comput Syst Sci 63(4):512–530

Kannan R (1987) Minkowski's convex body theorem and integer programming. Math Oper Res 12(3):415–440

Lenstra HW (1983) Integer programming with a fixed number of variables. Math Oper Res 8(4):538–548

Lovász L (1977) A homology theory for spanning trees of a graph. Acta Mathematica Academiae Scientiarum Hungarica 30:241–251

Lucertini M, Perl Y, Simeone B (1993) Most uniform path partitioning and its use in image processing. Discrete Appl Math 42(2):227–256

Maravalle M, Simeone B, Naldini R (1997) Clustering on trees. Comput Stat Data Anal 24(2):217–234

Miyazawa FK, Moura PFS, Ota MJ, Wakabayashi Y (2020) Cut and flow formulations for the balanced connected k-partition problem. In: Baïou M, Gendron B, Günlük O, Mahjoub AR (eds) Combin Optim. Springer, Cham, pp 128–139

Miyazawa FK, Moura PF, Ota MJ, Wakabayashi Y (2021) Partitioning a graph into balanced connected classes: formulations, separation and experiments. Eur J Oper Res 293(3):826–836

Moura PFS, Ota MJ, Wakabayashi Y (2022) Approximation and parameterized algorithms for balanced connected partition problems. In: Balachandran N, Inkulu R (eds) Algorithms and discrete applied mathematics, Lecture notes in computer science, vol 13179. Springer, Berlin, pp 211–223

Nakano S, Rahman M, Nishizeki T (1997) A linear-time algorithm for four-partitioning four-connected planar graphs. Inf Process Lett 62(6):315–322

Perl Y, Schach SR (1981) Max-min tree partitioning. J ACM 28(1):5–15

Suzuki H, Takahashi N, Nishizeki T (1990a) A linear algorithm for bipartition of biconnected graphs. Inf Process Lett 33(5):227–231

Suzuki H, Takahashi N, Nishizeki T, Miyano H, Ueno S (1990b) An algorithm for tripartitioning 3-connected graphs. J Inf Process Soc Jpn 31(5):584–592

Wu BY (2012) Fully polynomial-time approximation schemes for the max–min connected partition problem on interval graphs. Discrete Math Algorithms Appl 04(01):1250005

Zhou X, Wang H, Ding B, Hu T, Shang S (2019) Balanced connected task allocations for multi-robot systems: an exact flow-based integer program and an approximate tree-based genetic algorithm. Expert Syst Appl 116:10–20