Int J Data Sci Anal (2016) 1:17–35
DOI 10.1007/s41060-016-0005-2

REGULAR PAPER

Bounded correlation clustering

Floris Geerts[1] · **Reuben Ndindi**[1]

Received: 22 January 2016 / Accepted: 29 January 2016 / Published online: 16 February 2016

Abstract Correlation clustering is to partition a set of objects into clusters such that the number of false positives and negatives is minimized. In this paper, we combine correlation clustering and user interaction. More specifically, we allow the user to control the quality of the clustering by providing error bounds on the number of false positives and negatives. If no clustering exists that satisfies these bounds, a set of edges is returned for user inspection such that the deletion or relabeling of these edges guarantees the existence of a clustering consistent with the error bounds. A user, however, may reject the deletion or relabeling of certain edges and ask for an alternative set of edges to be provided. If no such set of edges exists, a minimal change to the error bounds should be provided, after which the interactive process continues. The focus of this paper is on the algorithmic challenges involved in returning a minimal set of edges to the user. More specifically, we formalize the BOUNDED CORRELATION CLUSTERING problem and show that it is intractable. Therefore, we propose an approximation algorithm based on the well-known region growing technique. We experimentally validate the efficiency and accuracy of the approximation algorithm.

Keywords Clustering · Multicut · Approximation algorithms · User interaction

✉ Floris Geerts
floris.geerts@uantwerpen.be

Reuben Ndindi
reuben.ndindi@uantwerpen.be

[1] Department of Mathematics and Computer Science, University of Antwerp, 2020 Antwerp, Belgium

1 Introduction

Clustering is to partition a set of given objects into clusters of similar objects. Typically, the goal is to find a clustering that minimizes an objective function that measures the quality of the clustering. A wide variety of formalizations and objective functions have been considered in this context. We refer the reader to [1] for a comprehensive survey on clustering techniques in data mining and machine learning.

In this paper, we focus on the formalization of the clustering problem, known as CORRELATION CLUSTERING [2]. Intuitively, in correlation clustering the set of objects are vertices of a graph whose edges are labeled with either "+" or "−". Here, a +-edge indicates that its vertices (objects) are similar, whereas a −-edge indicates the opposite. The corresponding objective function counts the number of *false positives*, i.e., − edges whose vertices belong to the same cluster, and the number of *false negatives*, i.e., + edges whose vertices belong two distinct clusters.

Correlation clustering can be used in applications such as protein interaction networks [4], crosslingual link detection [17], communication networks [3], among others. It provides a unique approach in solving problems where we have conflicting measures among objects and the aim is to provide a consistent clustering.

Although one of the nice features of correlation clustering is that a user does not need to specify anything, this is at the same time also one of its shortcomings. Indeed, a user does not have any control on the quality of the clustering returned by correlation clustering. In this paper, we therefore revisit correlation clustering in the presence of user specified *error bounds*.

More specifically, we envisage a clustering system where a user should be able to interact with the system by indicating his/her preferences in terms of clustering errors thereby

impacting the quality of the clustering. For instance, a user may want to express that a clustering should not have any errors, or more generally, want to bound the number of false positives and negatives in a clustering. Such a setting is of interest to any application domain of correlation clustering as it gives the user the possibility to control the quality of the clustering by mean of these error bounds.

To bring user interaction into correlation clustering is quite challenging, however. Indeed, suppose that a user specifies two *error bounds*, μ_{fp} and μ_{fn}, for the false positives and negatives, respectively. With these bounds, she/he expresses that a clustering is desired in which the number of false positives and negatives does not exceed the given bounds. We call such a clustering a *valid* clustering. Unfortunately, a valid clustering may not exist. We therefore generalize the correlation clustering problem to the bounded correlation problem.

In *bounded correlation clustering*, we want to guide the user toward a valid clustering by allowing him/her to either minimally update the graph, to minimally change the error bounds, or combinations thereof. For example, one way to guarantee the existence of a valid clustering is to *delete* or *relabel* a set ΔE of edges. Indeed, deleting all edges or assigning all edges the same label trivially guarantees such valid clusterings.

Of course, we want to minimally modify the input graph. Therefore, we envisage a bounded correlation clustering system that provides the user with a *minimal* set ΔE of edges to delete/relabel. We provide a detailed motivating example in the next section illustrating user interaction with the bounded correlation clustering system.

In this paper, we focus on one key algorithmic component of the interactive framework: when given an input graph G, error bounds μ_{fp} and μ_{fn}, return a set ΔE of edges to the user such that the deletion/relabeling of edges in ΔE guarantees the existence of a valid clustering, i.e., a clustering of the updated graph that satisfies the error bounds. More specifically, we make the following contributions:

- We formally define the BOUNDED CORRELATION CLUSTERING problem, and we show that it is intractable.
- We present a region growing-based approximation algorithm for solving the BOUNDED CORRELATION CLUSTERING problem and provide a performance guarantee. More specifically, the size of the set of edges returned by the algorithm is at most a factor $O(\log(|E|^-))$ away from the optimal size. Here, E^- represents the set of "−" labeled edges in the input graph. The approximation algorithm leverages a close relationship between BOUNDED CORRELATION CLUSTERING and a variant of the MULTICUT problem, called BOUNDED MULTICUT problem, which may be of interest in its own right.
- We show how our algorithms could be used in the context of interactive settings such as illustrated in Sect. 2.
- We empirically evaluate our algorithm on both synthetic and real-life datasets. Although the approximation factor may be large in theory, we verify that in practice, the size of the returned set of edges is close to optimal.

This work extends [11] by dealing with both deletions and relabeling of edges (Sect. 3) and by describing how our algorithm for the BOUNDED CORRELATION problem can be used in the context of a user interaction process (Sect. 6). More importantly, the approximation algorithm reported in [11] is substantially improved. In particular, our new algorithm **BMulticut** now implements an "adaptive" optimization strategy in which the underlying integer program (and its relaxation) is updated between two region growing steps provided that this is expected to lead to a better solution (Sect. 5). Furthermore, we provide a thorough analysis of the algorithm for correctness, which was not present in [11] and obtain an approximation guarantee matching our previous algorithm, in the worst case, and improves on our previous algorithm in most cases. These analyses are more challenging than for the algorithm given in [11]. Furthermore, we have revised and included additional experiments on real and synthetic datasets (Sect. 7). Improvements on the quality of approximate solutions have been observed in comparison to the algorithm given in [11].

Organization of the paper. The rest of the paper is organized as follows. In the next section, we provide a detailed example of an interactive clustering process. In Sect. 3, we formally define the BOUNDED CORRELATION CLUSTERING problem and establish its intractability. In Sect. 4, we introduce the BOUNDED MULTICUT problem and establish its relationship with the BOUNDED CORRELATION CLUSTERING problem. Our region growing-based approximation algorithm is presented in Sect. 5. We provide further details on how the algorithm fits into our proposed interactive framework in Sect. 6. An experimental evaluation on both synthetic and real-life data is presented in Sect. 7. Related work is presented in Sect. 8. We conclude the paper in Sect. 9.

2 Motivating example

In this section we further illustrate the need for revising the standard correlation clustering problem in an interactive setting. More specifically, we demonstrate an interactive process in which user actions include providing error bounds for the false positives and negatives. Our clustering system will in return output a valid clustering if it exists, and otherwise return a minimal set of edges to be deleted/relabeled for user inspection. A user may additionally mark edges as immutable, ensuring that these edges will never be deleted or relabeled. Furthermore, feedback on revised error bounds is given in the case where the problem is over-constrained

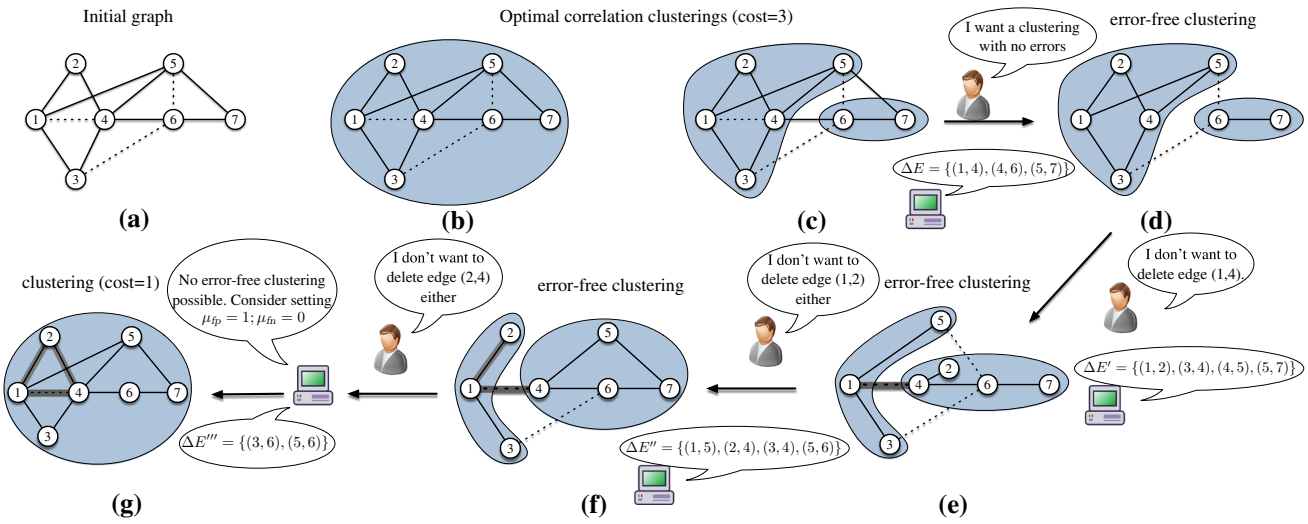


Fig. 1 Illustration of user interaction with the bounded correlation clustering process as explained in Example 1. In these figures *solid lines* represent $+$-labeled edges, *dashed lines* represent $-$ edges, immutable edges are thickened, and clusters are represented by *gray-shaded areas*

due the fact that a user marked too many edges as being immutable.

Example 1 Consider the graph G shown in Fig. 1a. It can be easily verified, e.g., by solving an integer program that encodes the correlation clustering problem [8], that an optimal correlation clustering of G will always have a total of three false positive and negatives. For example, the clustering shown in Fig. 1b has three false positives; the clustering shown in Fig. 1c has one false positive and two false negatives. However, suppose that a user wants a clustering with no errors and therefore sets the two *error bounds*, μ_{fp} and μ_{fn}, for the false positives and negatives, respectively, to zero. As previously noted, a valid clustering (a clustering with no errors) may not exist. Indeed, if $\mu_{\text{fp}} = \mu_{\text{fn}} = 0$, then G does not have a valid clustering as we just have seen that any optimal clustering of G has cost 3.

It is here that *bounded correlation clustering* comes into play and a set ΔE of edges, to be deleted or relabeled, is returned for user inspection. The deletion/relabeling of these edges guarantees the existence of a valid clustering. For example, deleting or relabeling the three edges $\Delta E = \{(1,4),(4,6),(5,7)\}$ corresponding to the errors in the clustering shown in Fig. 1c ensures that a valid clustering with no errors exists, as shown in Fig. 1d.

Furthermore, when presented with the set ΔE of edges, the user may decide not to delete/relabel an edge in ΔE since she/he regards the similarity information represented by this edge as too important or trustworthy. In this case, a user may mark such edges as *immutable*. The immutable edges are then passed on to the bounded correlation clustering system and another set of edges $\Delta E'$ is returned, which *excludes* the immutable edges. For example, the user may mark the edge $(1,4)$ as immutable. By fixing the edge $(1,4)$, one now has to delete a set of edges $\Delta E' = \{(1,2),(3,4),(4,5),(5,7)\}$ in order to obtain a valid clustering for the bounds $\mu_{\text{fp}} = \mu_{\text{fn}} = 0$. Figure 1e shows a valid clustering on the updated graph.

The user again inspects this set of edges and the interactive process continues until either the user is satisfied and a valid clustering does exist, or no valid clustering exists. The latter case happens when the user marked too many edges as immutable and no ΔE exists whose deletion/relabeling ensures a valid clustering.

For example, suppose that the user marks also the edge $(1,2)$ in G as immutable. Then to satisfy the bounds, a set of edges $\Delta E'' = \{(1,5),(2,4),(3,4),(5,6)\}$ needs to be deleted/relabeled. The corresponding valid clustering for $\mu_{\text{fp}} = \mu_{\text{fn}} = 0$ is shown in Fig. 1f. Imagine that at this point the user is still not happy with $\Delta E''$ and also marks the edge $(2,4)$ as immutable. In this case, no ΔE exists that guarantees a valid clustering.

To see why no valid clustering exists, consider that by enforcing the last user action i.e., edge $(2,4)$ is marked as immutable, the user requires an error-free clustering of a graph in which the triangle $(1,4)$, $(1,2)$ and $(2,4)$ is immutable. Since edges $(1,2)$ and $(2,4)$ are $+$-labeled, and $(1,4)$ is $-$-labeled, *any* clustering will induce at least one error due to presence of this triangle. Therefore, as long as this triangle remains immutable, no valid error-free clustering can exist.

Instead, the bounded correlation clustering system should inform the user as to how to minimally change the error bounds, for example, by letting $\mu_{\text{fp}} = 1$ and $\mu_{\text{fn}} = 0$ (see Fig. 1g). We will see in Sect. 6 how these revised error bounds can be obtained.

The interactive process then continues. Observe that this process always terminates. In the worst case, all edges in G are marked as immutable and the bounds μ_{fp} and μ_{fn} are set such that a valid optimal correlation clustering in G exists. For example, for $\mu_{\text{fp}} = 1$ and $\mu_{\text{fn}} = 2$ the clustering shown in Fig. 1c is valid. Note that in this case $\Delta E = \emptyset$.

The example thus clearly shows the need for considering bounded correlation clustering. In the next section, we formally introduce the BOUNDED CORRELATION CLUSTERING problem (Sect. 3.1) and establish its intractability (Sect. 3.2).

3 The bounded correlation clustering problem

We first introduce some concepts and notations.

Let $G = (V, E, w)$ be a graph with a weight function $w : E \to \mathbb{N}$ on its edges. Assume that the set E of edges can be partitioned into two sets E^+ and E^-. An edge $e \in E^+$ carries label "+", whereas an edge $e \in E^-$ carries label "−". Intuitively, edges in E^+ represent similar objects that should be clustered together; edges in E^- represent the opposite. In the following, we write $G = (V, E^+ \cup E^-, w)$ to make the partition of the edge set E explicit. For an edge $(u, v) \in E$ we interchangeably use w_{uv} and $w(u, v)$ to denote the weight of edge (u, v) relative to the weight function w.

A clustering $\mathcal{C}$ of G is a partition of V. Each partition in $\mathcal{C}$ is called a cluster. For a vertex $v \in V$, we denote by $\mathcal{C}(v)$ the set of vertices in the same cluster as v. In a clustering $\mathcal{C}$, we call an edge $e = (u, v)$ a false negative if $e \in E^+$ but $u \notin \mathcal{C}(v)$. In other words, a false negative is a +-labeled edge that crosses clusters. Similarly, if $e \in E^-$ and $u \in \mathcal{C}(v)$, we call $e = (u, v)$ a false positive. In other words, a false positive is a −-labeled edge within the same cluster.

We denote by $w_{\text{fn}}(\mathcal{C})$ and $w_{\text{fp}}(\mathcal{C})$ the sum of the weights of false negatives and positives in $\mathcal{C}$, respectively. Similarly, for an arbitrary set E of edges we define $w(E)$ as the sum of the weights of edges in E. Finally, we define $cost(\mathcal{C}) = w_{\text{fp}}(\mathcal{C}) + w_{\text{fn}}(\mathcal{C})$. Standard CORRELATION CLUSTERING is to find a clustering $\mathcal{C}$ of G such that $cost(\mathcal{C})$ is minimal.

We omit the set of immutable edges IE for the moment as these can be readily incorporated as shown in Sect. 6.

3.1 Problem statement

As already described in the Introduction, we want to control the errors (false positives and false negatives) that a clustering makes. Let μ_{fp} and μ_{fn} be two natural numbers. We regard a clustering $\mathcal{C}$ as being *valid* provided that $w_{\text{fp}}(\mathcal{C})$ and $w_{\text{fn}}(\mathcal{C})$ are below these thresholds.

We have seen in Example 1 that such valid clusterings do not always exist. The existence of valid clustering can be guaranteed, however, when sufficiently many edges are deleted/relabeled in the input graph. Clearly, we want to delete/relabel as less edges as possible. In the following we denote by ΔE_{d} the set of edges to be deleted and by ΔE_{r} the set of edges to be relabeled.

Problem 1 (BOUNDED CORRELATION CLUSTERING) Given a graph $G = (V, E^+ \cup E^-, w)$ and natural numbers μ_{fp} and μ_{fn}, find a set $\Delta E = \Delta E_{\text{d}} \cup \Delta E_{\text{r}}$ of edges, such that $w(\Delta E)$ is minimal and such that there exists a clustering $\mathcal{C}$ of the updated graph

$$G' = (V, (((E^+ \cup (\Delta E_{\text{r}} \cap E^-)) \cup (E^- \cup (\Delta E_{\text{r}} \cap E^+))) \setminus \Delta E_{\text{d}}, w),$$

for which $w_{\text{fn}}(\mathcal{C}) \leqslant \mu_{\text{fn}}$ and $w_{\text{fp}}(\mathcal{C}) \leqslant \mu_{\text{fp}}$ holds. □

Here, the updated graph G' is obtained from G as follows: (i) edges in E^+ are replaced by edges in $E^+ \cup (\Delta E_{\text{r}} \cap E^-)$, i.e., the set of +-labeled edges in E^+ together with those edges in ΔE_{r} that after relabeling become +-labeled edges as well; (ii) similarly edges in E^- are replaced by those in $E^- \cup (\Delta E_{\text{r}} \cap E^+)$; and finally (iii) edges in ΔE_{d} are deleted. We naturally assume that ΔE_{d} and ΔE_{r} are disjoint.

We next argue that we can focus on solutions of the form $\Delta E = (\Delta E_{\text{d}}, \emptyset)$,i.e., solutions in which only edges are to be deleted.

Proposition 1 *If* $\Delta E = (\Delta E_{\text{d}}, \Delta E_{\text{r}})$ *is a solution of* BOUNDED CORRELATION CLUSTERING *then for any edge* $e \in \Delta E_{\text{r}}$, $\Delta E' = (\Delta E_{\text{d}} \cup \{e\}, \Delta E_{\text{r}} \setminus \{e\})$ *is also a solution.*

Proof Let $\Delta E = (\Delta E_{\text{d}}, \Delta E_{\text{r}})$ be a solution of BOUNDED CORRELATION CLUSTERING and let $e \in \Delta E_{\text{r}}$. We show that $\Delta E' = (\Delta E_{\text{d}} \cup \{e\}, \Delta E_{\text{r}} \setminus \{e\})$ is a solution as well.

Let $\mathcal{C}$ be a valid clustering in the updated graph $G' = (V, (((E^+ \cup (\Delta E_{\text{r}} \cap E^-)) \cup (E^- \cup (\Delta E_{\text{r}} \cap E^+))) \setminus \Delta E_{\text{d}}, w)$. If e is a +-labeled edge in G' (and thus was a −-labeled edge in G) between clusters of $\mathcal{C}$ or e is a −-labeled edge in G' (and thus was a +-labeled edge in G) inside a cluster of $\mathcal{C}$, then deleting e results in the elimination of a false negative or false positive, respectively. Hence, $\mathcal{C}$ is a clustering of smaller cost in the graph obtained from G' by deleting edge e. Hence, $\Delta E' = (\Delta E_{\text{d}} \cup \{e\}, \Delta E_{\text{r}} \setminus \{e\})$ is a solution. Furthermore, if e is a −-labeled edge between clusters of $\mathcal{C}$ or a +-labeled edge inside a cluster of $\mathcal{C}$, then deleting e in G' does not affect the cost of $\mathcal{C}$ and hence $\Delta E' = (\Delta E_{\text{d}} \cup \{e\}, \Delta E_{\text{r}} \setminus \{e\})$ is clearly a solution as well. □

This proposition implies that, without loss of generality, we may concentrate on solutions of BOUNDED CORRELATION CLUSTERING in which edges are only deleted. In the following, when we write ΔE we mean $(\Delta E_{\text{d}}, \emptyset)$, unless specified otherwise.

3.2 Intractability result

Not surprisingly, the BOUNDED CORRELATION CLUSTERING problem is computationally infeasible. Indeed, its decision version that is to determine given $G = (V, E^+ \cup E^-, w)$, μ_{fp}, μ_{fn}, and integer $L \geq 0$ whether or not there exists a set ΔE of edges such that $w(\Delta E) \leqslant L$ and such that after deleting ΔE from G, the updated graph $G' = (V, (E^+ \cup E^-) \setminus \Delta E, w)$ has a clustering $\mathcal{C}$ such that $w_{\text{fp}}(\mathcal{C}) \leqslant \mu_{\text{fp}}$ and $w_{\text{fn}}(\mathcal{C}) \leqslant \mu_{\text{fn}}$, is NP-complete.

Proposition 2 *The decision version of* BOUNDED CORRELATION CLUSTERING *is NP-complete for both weighted and unweighted graphs.*

Proof For the lower bound, we prove that the decision version of BOUNDED CORRELATION CLUSTERING is NP-hard by reducing it from the decision version of CORRELATION CLUSTERING. The latter decision version is to determine given an input graph $H = (W, F^+ \cup F^-, w)$ and integer $K \geq 0$, whether or not there exists a clustering $\mathcal{C}$ of H such that $cost(\mathcal{C}) \leqslant K$. This problem was proven to be NP-hard in [2] for both weighted and unweighted graphs.

The reduction is as follows. Let $H = (W, F^+ \cup F^-, w)$ and $K \geq 0$ be an instance of CORRELATION CLUSTERING. We define the corresponding instance of BOUNDED CORRELATION CLUSTERING by letting $G = H$, $L = K$, $\mu_{\text{fn}} = 0$ and $\mu_{\text{fp}} = 0$.

For the correctness of the reduction, consider a clustering $\mathcal{C}$ of H such that $cost(\mathcal{C}) \leqslant K$. If we delete all edges corresponding to the false positives and negatives in $\mathcal{C}$ from H, then the clustering induced by $\mathcal{C}$ on the updated graph has no false positives and negatives. Hence, by letting ΔE be the set of edges corresponding to the false positives and negatives in $\mathcal{C}$, we obtain a solution for BOUNDED CORRELATION CLUSTERING with $|\Delta E| \leqslant L = K$, $\mu_{\text{fn}}(\mathcal{C}) = 0$ and $\mu_{\text{fp}}(\mathcal{C}) = 0$. Conversely, suppose that by deleting edges in ΔE from G with $|\Delta E| \leqslant K$, we have that there is a clustering $\mathcal{C}$ of $G' = (V, (E^+ \cup E^-) \setminus \Delta E, w)$ with no false positives and negatives. Then, $\mathcal{C}$ is a clustering of H such that $cost(\mathcal{C}) = w(\Delta E) \leqslant K = L$. Hence, solutions of CORRELATION CLUSTERING correspond to solutions of BOUNDED CORRELATION CLUSTERING with $\mu_{\text{fn}} = 0$ and $\mu_{\text{fp}} = 0$, and vice versa.

For the upper bound, consider the following NP-algorithm: (1) Guess (a) a set ΔE of at most L edges; and (b) a clustering $\mathcal{C}$ of the updated graph $G' = (V, (E^+ \cup E^-) \setminus \Delta E, w)$. (2) Verify (in PTIME) whether $w_{\text{fn}}(\mathcal{C}) \leqslant \mu_{\text{fn}}$ and $w_{\text{fp}}(\mathcal{C}) \leqslant \mu_{\text{fp}}$ hold. If so, accept the guess and return "yes"; otherwise reject the guess. Clearly, this algorithm correctly decides the (decision variant of) BOUNDED CORRELATION CLUSTERING. □

In view of this intractability result, we develop an approximation algorithm for the BOUNDED CORRELATION CLUSTERING problem. Our solution to the BOUNDED CORRELATION CLUSTERING problem is obtained by following a similar strategy as is used for the approximation algorithm for the CORRELATION CLUSTERING problem given in [8]. More specifically, we first establish a relationship between the BOUNDED CORRELATION CLUSTERING problem and a variant of the MULTICUT problem, called the BOUNDED MULTICUT problem. Next, we use a region growing technique for BOUNDED MULTICUT to obtain an approximation algorithm. Region growing was introduced in [10] as a technique for approximation the MULTICUT problem.

4 The bounded multicut problem

In this section, we establish a relationship between the BOUNDED CORRELATION CLUSTERING problem and a variant of the MULTICUT problem, called the BOUNDED MULTICUT problem (Sect. 4.1). This relationship ensures that an approximation algorithm for the BOUNDED MULTICUT problem automatically results in an approximation algorithm for the BOUNDED CORRELATION CLUSTERING problem (Observation 1 below). As previously mentioned, the approximation algorithm for the BOUNDED MULTICUT problem is based on the region growing technique [10]. This technique requires solving a linear relaxation of an integer program for the BOUNDED MULTICUT problem. We end this section by providing such an integer program for the BOUNDED MULTICUT problem and show its correctness (Sect. 4.2). As a side result, we obtain an exact algorithm for the BOUNDED MULTICUT problem and thus also for the BOUNDED CORRELATION CLUSTERING problem, due to their close relationship.

4.1 Bounded multicut and its relation to bounded correlation clustering

We start by introducing the BOUNDED MULTICUT problem. Recall that an instance of the standard MULTICUT problem consists of an edge-weighted graph $G = (V, E, w)$ together with a set $S = \{(s_i, t_i) \mid i \in [1, k]\}$ of source–sink pairs, and is to find a set T of minimal weight (a so-called *multicut*) such that the removal of the edges in T from G disconnects all pairs in S (See [13] for the first appearance of this problem).

The BOUNDED MULTICUT problem differs from MULTICUT in that it works on graphs whose edge set is partitioned in E^+ and E^-, just as in BOUNDED CORRELATION CLUSTERING; and bounds μ^+ and μ^- are present that limit the allowed number of positive and negative edges, respectively, in a multicut. A multicut T such that $w(T \cap E^+) \leqslant \mu^+$ and $w(T \cap E^-) \leqslant \mu^-$ is referred to as a *valid* multicut. Similarly as in BOUNDED CORRELATION CLUSTERING a valid multicut may not always exist, however. Hence, BOUNDED MULTICUT asks for a minimal set of edges to be deleted such

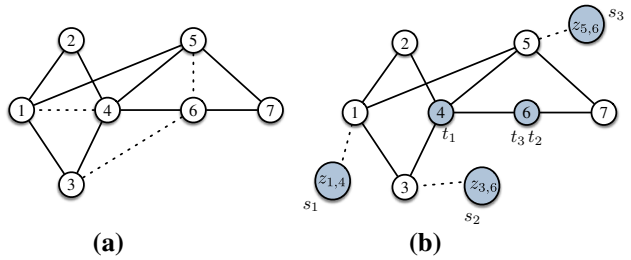


Fig. 2 Transformation from BOUNDED CORRELATION CLUSTERING to BOUNDED MULTICUT

that the existence of a valid multicut is guaranteed. More formally,

Problem 2 (BOUNDED MULTICUT) Given a graph $G = (V, E^+ \cup E^-, w)$, a set S of pairs of distinct vertices (s_i, t_i) of G, and bounds μ^+ and μ^-, find a set of edges ΔE of minimal weight such that there is a multicut T in $G = (V, (E^+ \cup E^-) \setminus \Delta E, w)$ such that $w(T \cap E^+) \leqslant \mu^+$ and $w(T \cap E^-) \leqslant \mu^-$ □

In analogy to the relationship between MULTICUT and CORRELATION CLUSTERING, as described in [8], we present a procedure for transforming an instance of BOUNDED CORRELATION CLUSTERING to an instance of BOUNDED MULTICUT.

Let $G = (V, E^+ \cup E^-, w)$ be an input graph of the BOUNDED CORRELATION CLUSTERING problem with bounds μ_{fp} and μ_{fn}. We obtain the corresponding BOUNDED MULTICUT instance $G_{\text{bmc}} = (V_{\text{bmc}}, E^+_{\text{bmc}} \cup E^-_{\text{bmc}}, w_{\text{bmc}})$ with bounds μ^+ and μ^- and set of source–sink pairs S, as follows:

(1) For every edge $(u, v) \in E^-$, we introduce a new vertex $z_{u,v}$. We define V_{bmc} as V together with these newly added vertices.
(2) We define $E^+_{\text{bmc}} = E^+$ and let E^-_{bmc} consist of one new $-$-labeled edge $(z_{u,v}, u)$ for each vertex of the form $z_{u,v}$. The weight of such an edge $w_{\text{bmc}}(z_{u,v}, u)$ is set to $w(u, v)$. Furthermore, $w_{\text{bmc}}(u, v) = w(u, v)$ for every $(u, v) \in E^+_{\text{bmc}}$.
(3) We let S consists of the source–sink pairs $(z_{u,v}, v)$, for newly added vertices $z_{u,v}$.
(4) We set $\mu^+ = \mu_{\text{fn}}$ and $\mu^- = \mu_{\text{fp}}$.

This transformation only differs from the transformation presented in [8] in that the edges $(z_{u,v}, u)$ are given a $-$-label and the edges in E^+ retain their $+$-label. By contrast, [8] translates an instance of CORRELATION CLUSTERING into an instance MULTICUT in which no $+$-or $-$-labels are present.

Example 2 Consider the graph G from Example 1, which is shown again in Fig. 2a for convenience. The BOUNDED MULTICUT instance G_{bmc} corresponding to G is shown in Fig. 2b. According to the transformation just described, the BOUNDED MULTICUT instance includes three new vertices $z_{1,4}$, $z_{3,6}$, $z_{5,6}$, corresponding to the $-$-labeled edges $(1, 4)$, $(3, 6)$ and $(5, 6)$ in G, respectively. Furthermore, G_{bmc} contains three new $-$-labeled edges replacing the original negative edges. That is, $E^-_{\text{bmc}} = \{(z_{1,4}, 1), (z_{3,6}, 3), (z_{5,6}, 5)\}$. Finally, the set S of source–sink pairs (shaded vertices in Fig. 2b) are consists of $(s_1, t_1) = (z_{1,4}, 4)$, $(s_2, t_2) = (z_{3,6}, 6)$, and $(s_3, t_3) = (z_{5,6}, 6)$. □

We now show that solutions of the BOUNDED CORRELATION CLUSTERING problem for G with bounds μ_{fp} and μ_{fn} correspond to solutions of the BOUNDED MULTICUT PROBLEM for G_{bmc}, S and bounds $\mu^+ = \mu_{\text{fn}}$ and $\mu^- = \mu_{\text{fp}}$, and vice versa. We first show this equivalence for clusterings and multicuts.

Lemma 1 *A valid clustering of $\mathcal{C}$ in $G = (V, E^+ \cup E^-, w)$ relative to the bounds μ_{fn} and μ_{fp} corresponds to a valid multicut T in $G_{\text{bmc}} = (V_{\text{bmc}}, E^+_{\text{bmc}} \cup E^-_{\text{bmc}}, w_{\text{bmc}})$, S and bound μ^+ and μ^-, and vice versa.*

Proof The proof is analogous to Lemmas 4.5 and 4.6 in [8]. Let $\mathcal{C}$ be a valid clustering $G = (V, E^+ \cup E^-, w)$ and let T consist of all edges in $E^+_{\text{bmc}} = E^+$ that contribute to $w_{\text{fn}}(\mathcal{C})$, and all edges $(z_{u,v}, u) \in E^-_{\text{bmc}}$ that correspond to an edge $(u, v) \in E^-$ that contribute to $w_{\text{fp}}(\mathcal{C})$. It is shown in [8] that T is a multicut. Furthermore, observe that when $\mathcal{C}$ is valid relative to μ_{fn} and μ_{fp}, then $w(T \cap E^+_{\text{bmc}}) \leqslant \mu^+$ and $w(T \cap E^-_{\text{bmc}}) \leqslant \mu^-$. Hence, T is a valid multicut.

Similarly, given a valid multicut T in G_{bmc}, one can construct a clustering $\mathcal{C}$ as follows. Let T' be union of the set of edges in $E^+ \cap T$ (recall that $E^+ = E^+_{\text{bmc}}$) and the set of edges $(u, v) \in E^-$ corresponding to an edge $(z_{u,v}, u) \in E^-_{\text{bmc}} \cap T$. Denote by G^+ the graph G restricted to $+$-labeled edges. Then, $\mathcal{C}$ is defined as the set of all connected components in $G^+ \setminus T$. It is shown in [8] that T' consist of the false positives and negatives of the clustering $\mathcal{C}$. Hence, $w_{\text{fp}}(\mathcal{C}) \leqslant \mu^-$ and $w_{\text{fn}}(\mathcal{C}) \leqslant \mu^+$. In other words, $\mathcal{C}$ is a valid clustering of G. □

Example 3 We next illustrate Lemma 1. Recall the transformation shown in Fig. 2. Consider the valid clustering $\mathcal{C}$ of G for $\mu_{\text{fp}} = 2$ and $\mu_{\text{fn}} = 3$ as shown in Fig. 3a. Here, $w_{\text{fn}}(\mathcal{C})$ and $w_{\text{fp}}(\mathcal{C})$ correspond to the false negatives $\{(1, 5), (4, 5), (4, 6)\}$ and false positives $\{(1, 4), (5, 6)\}$, respectively. To obtain a valid multicut T for $\mu^+ = 3$ and $\mu^- = 2$ in G_{bmc}, we simply define $T = \{(1, 5), (4, 5), (4, 6), (z_{1,4}, 1), (z_{5,6}, 5)\}$ as shown in Fig. 3b, where $z_{1,4}$ and $z_{5,6}$ denote the newly added vertices in the transformation from G to G_{bmc}. Conversely, given the valid multicut T of G_{bmc}, for $\mu^+ = 3$ and $\mu^- = 2$, as shown in Fig. 3b, we obtain the valid clustering of G as shown in Fig. 3a by declaring the positive edges in T as false negatives

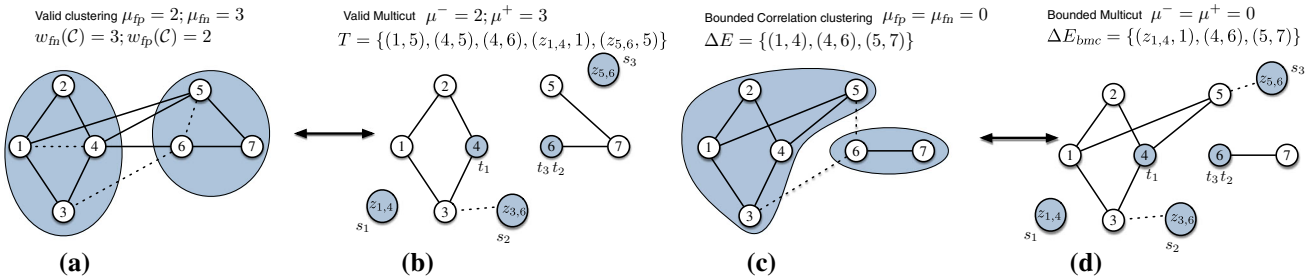


Fig. 3 **a, b** Going from clusterings to multicuts, and back; **c, d** going from solutions of BOUNDED CORRELATION CLUSTERING to BOUNDED MULTICUT, and back

and the edges $(1, 4)$ and $(5, 6)$ in G corresponding to the negative edges $(z_{1,4}, 1)$ and $(z_{5,6}, 5)$, respectively, in T as false positives. The connected components in $G^+ \setminus T$ constitute the clusters in the valid clustering $\mathcal{C}$ of G. □

Consider now an instance $G = (V, E^+ \cup E^-, w)$ of BOUNDED CORRELATION CLUSTERING with bounds μ_{fp} and μ_{fn}. Let ΔE be a solution. That is, there exists a valid clustering $\mathcal{C}$ on the updated graph $G' = (V, (E^+ \cup E^-) \setminus \Delta E, w)$. We know from Lemma 1 that $\mathcal{C}$ corresponds to a valid multicut T in the BOUNDED MULTICUT instance $(G')_{\text{bmc}}$,i.e., the instance obtained by applying the transformation on the updated instance G'. It is now easily verified that $(G')_{\text{bmc}}$ is equal to the BOUNDED MULTICUT instance obtained from deleting a set of edges ΔE_{bmc} from G_{bmc}. More specifically, $\Delta_{\text{bmc}} E$ consist of $\Delta E \cap E^+_{\text{bmc}}$ and all edges $(z_{u,v}, u) \in E^-_{\text{bmc}}$ for which $(u, v) \in \Delta E \cap E^-$. Note that $\Delta E \cap E^+_{\text{bmc}}$ is well-defined since $E^+_{\text{bmc}} = E^+$ by construction. Furthermore, $w_{\text{bmc}}(\Delta E_{\text{bmc}}) = w(\Delta E)$. Hence, ΔE_{bmc} is a solution of the BOUNDED MULTICUT instance G_{bmc}, S, μ^+ and μ^-.

Conversely, consider a solution ΔE_{bmc} of the BOUNDED MULTICUT instance $G_{\text{bmc}} = (V_{\text{bmc}}, E^+_{\text{bmc}} \cup E^-_{\text{bmc}}, w_{\text{bmc}})$, S and bounds μ^+ and μ^-. In other words, there exists a valid multicut T in the updated graph $(G_{\text{bmc}})' = (V_{\text{bmc}}, (E^+_{\text{bmc}} \cup E^-_{\text{bmc}}) \setminus \Delta E_{\text{bmc}}, w_{\text{bmc}})$. It is readily verified that $(G_{\text{bmc}})' = (G')_{\text{bmc}}$ where G' is obtained from deleting a set ΔE of edges from G. Indeed, ΔE consists of $\Delta E_{\text{bmc}} \cap E^+$ and edges $(u, v) \in E^-$ corresponding to an edge $(z_{u,v}, u) \in \Delta E_{\text{bmc}} \cap E^-_{\text{bmc}}$. By Lemma 1, we know that T corresponds to a valid clustering $\mathcal{C}$ in G'. Observe that $w(\Delta E) = w_{\text{bmc}}(\Delta E_{\text{bmc}})$. Hence, ΔE is a solution of the BOUNDED CORRELATION CLUSTERING instance G, μ_{fp} and μ_{fn}.

Example 4 Figures 3c, d illustrate the correspondence between solutions ΔE of the bounded correlation clustering problem for G, $\mu_{\text{fp}} = \mu_{\text{fn}} = 0$, and solutions ΔE_{bmc} of the bounded multicut problem for G_{bmc}, $\mu^+ = \mu^- = 0$ and $S = \{(z_{1,4}, 4), (z_{3,6}, 6), (z_{5,6}, 6)\}$. The relationship between these two is just as described in Example 3. □

An immediate consequence of the equivalence between the two problems is the following.

Observation 1 *Any (approximation) algorithm for the* BOUNDED MULTICUT *problem results in an (approximation) for the* BOUNDED CORRELATION CLUSTERING *problem.*

In other words, the approximation algorithm for BOUNDED MULTICUT that will be presented in the next section is indeed an approximation algorithm for BOUNDED CORRELATION CLUSTERING.

4.2 An exact solution for the bounded multicut problem

We show that the BOUNDED MULTICUT problem can be solved exactly by means of an integer program. The relaxation of this program will be used in the approximation algorithm in Sect. 5.

The integer program for BOUNDED MULTICUT, denoted by IP_{BMC} and shown below, is a modification of the standard program for solving MULTICUT [18]. Given a graph $G = (V, E^+ \cup E^-, w)$, set S of k distinct source–sink pairs $\{(s_i, t_i) \mid i \in [1, k]\}$, and natural numbers μ^+ and μ^- as input for BOUNDED MULTICUT, the corresponding integer program IP_{BMC} is given by:

IP_{BMC}: minimize $\sum_{(u,v)\in E} w_{uv}(x_{uv} - y_{uv})$
subject to

$$\sum_{(u,v)\in p_i} x_{uv} \geq 1,\ p_i \in \mathcal{P}_i,\ 1 \leq i \leq k \quad \text{(i)}$$
$$\sum_{(u,v)\in E^+} w_{uv} y_{uv} \leq \mu^+ \quad \text{(ii)}$$
$$\sum_{(u,v)\in E^-} w_{uv} y_{uv} \leq \mu^- \quad \text{(iii)}$$
$$x_{uv} \geq y_{uv} \quad \text{(iv)}$$
$$x_{uv}, y_{uv} \in \{0, 1\} \quad \text{(v)}$$

Here, $\mathcal{P}_i$ denotes the set of all paths from s_i to t_i in G, for $i \in [1, k]$. Observe that this integer program has exponentially many constraints but, similarly as in the standard MULTICUT case, it can be converted into one of polynomial size. For completeness, we provide this conversion below. Furthermore, note that the integer program for the standard MULTICUT problem, denoted by IP_{MC} and given e.g., in [18], can be obtained by setting μ^+ and μ^- to zero, i.e., by ignoring the y_{uv} variables.

We first verify the correctness of the integer program IP_{BMC}.

Proposition 3 *A solution of IP_{BMC} corresponds to a solution of the* BOUNDED MULTICUT *problem, and vice versa.*

Proof Let ΔE be a solution of BOUNDED MULTICUT and let T be a multicut in $G = (V, (E^+ \cup E^-) \setminus \Delta E, w)$ such that $w(T \cap E^+) \leqslant \mu^+$ and $w(T \cap E^-) \leqslant \mu^-$. Based on this, we define the following valuation ν:

$$\nu(x_{uv}) = \begin{cases} 1 & \text{if } (u, v) \in T \cup \Delta E \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\nu(y_{uv}) = \begin{cases} 1 & \text{if } (u, v) \in T \\ 0 & \text{otherwise.} \end{cases}$$

We claim that this valuation satisfies the conditions (i)–(v) of the integer program IP_{BMC}. Clearly, (iv) and (v) are satisfied by the definition. Note that $T \cup \Delta E$ is a multicut for the original graph $G = (V, E^+ \cup E^-, w)$ and thus any path from a source to a sink in S passes through $T \cup \Delta E$. Since $\nu(x_{uv}) = 1$ for all $(u, v) \in T \cup \Delta E$, condition (i) is satisfied. Clearly, (ii) and (iii) are satisfied since $\nu(y_{uv}) = 1$ for all $(u, v) \in T$ and T is a valid multicut with regard to the bounds μ^+ and μ^-. Finally, we also remark that the objective function corresponds to $w(\Delta E) = \sum_{e \in \Delta E} w_e$. Indeed, $\nu(x_{uv}) - \nu(y_{uv}) = 1$ for all $(u, v) \in \Delta E$, and $\nu(x_{uv}) - \nu(y_{uv}) = 0$ for all other edges. Because ΔE is a solution, $w(\Delta E)$ is minimal and the valuation ν minimizes the objective function of IP_{BMC}.

For the converse, let ν be a valuation that satisfies conditions (i)–(v). Consider the set of edges $\Delta E = \{(u, v) \mid \nu(x_{uv}) - \nu(y_{uv}) = 1\}$ and let $T = \{(u, v) \mid \nu(y_{uv}) = 1\}$. It can be readily verified that this results in a solution of BOUNDED MULTICUT. Indeed, it is known that condition (i) expresses that every source–sink pair is disconnected [18]. As a consequence, $T \cup \Delta E$ is multicut in G. Furthermore, conditions (ii) and (iii) imply that $w(T \cap E^+) \leqslant \mu^+$ and $w(T \cap E^-) \leqslant \mu^-$, respectively. As before, $w(\Delta E) = \sum_{e \in \Delta E} w_e$ and since ν minimizes the objective function of IP_{BMC}, ΔE has minimal weight as well. □

We next describe a standard procedure to turn IP_{BMC} into an equivalent integer program of polynomial size [10]. This is important to guarantee that our approximation algorithm runs in polynomial time, as we will see in the next section.

Let S be the set of k source–sink pairs. We introduce binary variables z_u^i, one for each vertex u in the graph and each $(s_i, t_i) \in S$. We then replace the constraint (i) in IP_{BMC} with the following two constraints:

$$z_u^i - z_v^i \leqslant x_{uv}, \quad \text{for all } (u, v) \in E^+ \cup E^-, 1 \leq i \leq k \ (\text{i}') \\ z_{s_i}^i - z_{t_i}^i \geq 1, \quad \text{for all } (s_i, t_i) \in S. \qquad (\text{i}'')$$

We show that constraint (i) is equivalent to the constraints (i′) and (i″). Consider a source–sink pair (s_i, t_i) in S and assume that we have a path p between this pair consist of the following edges $(s_i, v_1), (v_1, v_2), \ldots, (v_n, t_i)$.

Using constraint (i″) we have that

$$1 \leqslant z_{s_i}^i - z_{t_i}^i = z_{s_i}^i - z_{v_1}^i + z_{v_1}^i - z_{v_2}^i + \cdots - z_{v_n}^i + z_{v_n}^i - z_{t_i}^i$$

and using constraint (i′), we have

$$z_{s_i}^i - z_{v_1}^i \leqslant x_{s_i v_1}, z_{v_1}^i - z_{v_2}^i \leqslant x_{v_1 v_2}, \ldots, z_{v_n}^i - z_{t_i}^i \leqslant x_{v_n t_i}.$$

Hence,

$$1 \leqslant z_{s_i}^i - z_{t_i}^i \leqslant \sum_{(u,v) \in p} x_{uv}.$$

Note that this holds for any path p between any source–sink pairs. Hence, constraint (i) is satisfied.

For the converse, assume that constraint (i) is satisfied. For each $(s_i, t_i) \in S$ we set the variables z_u^i as follows: $z_u^i = \sum_{(u', w') \in p} x_{u'w'}$ where p is the shortest path from u to t_i. In particular, this implies that $z_{s_i}^i \geqslant 1$ by constraint (i) and $z_{t_i}^i = 0$ since t_i lies at distance 0 from t_i. Hence, (i″) is satisfied. Furthermore, note that for an edge $(u, v) \in E$ we have that $z_u^i - z_v^i = x_{uv}$ hence (i′) is satisfied as well.

We may thus conclude that replacing constraint (i) with the constraints (i′) and (i″) results in an equivalent integer program formulation of BOUNDED MULTICUT, of polynomial size.

5 The BMulticut approximation algorithm

In this section we describe the **BMulticut** approximation algorithm for the BOUNDED MULTICUT problem. As we have just observed, **BMulticut** will also be an approximation algorithm for the BOUNDED CORRELATION CLUSTERING problem, by leveraging the translation between the two problems (cfr. Observation 1).

The **BMulticut** algorithm is a modification of the standard region growing algorithm for the MULTICUT problem as given in [10]. Intuitively, the standard region growing algorithm solves a single linear program,i.e., the relaxation of the integer program IP_{MC} that encodes the MULTICUT problem. Next, the algorithm repeatedly grows regions until all source–sink pairs are disconnected. The edges adjacent to the regions then constitute a multicut T.

By contrast, **BMulticut** uses the relaxation of the integer program IP_{BMC} that encodes the BOUNDED MULTICUT problem. Furthermore, **BMulticut** implements an "adaptive" optimization strategy in which the underlying integer program (and its relaxation) is updated between two consecutive

BMulticut $(G_0, S_0, \mu_0^+, \mu_0^-, K)$
1. $\mathbf{d}_0 := \mathbf{SolveLP}(G_0, S_0, \mu_0^+, \mu_0^-)$;
2. $(G_1, S_1, B_0) := \mathbf{GRegion}(G_0, S_0, \mathbf{d}_0, |S_0|, 1)$;
3. $\mu_1^+ := \max\{0, \mu_0^+ - w(\partial B_0 \cap E_0^+)\}$;
   $\mu_1^- := \max\{0, \mu_0^+ - w(\partial B_0 \cap E_0^-)\}$;
4. $i := 1$; next :=true;
5. **While** $((\mu_i^+ \neq \mu_{i-1}^+$ **or** $\mu_i^- \neq \mu_{i-1}^-)$ **and** $|S_i| > 0$ **and** $i \leqslant K$ **and** next) **do**
6. $\quad \mathbf{d}_i := \mathbf{SolveLP}(G_i, S_i, \mu_i^+, \mu_i^-)$;
7. $\quad$ **If** $\sum_{e \in E_i^+ \cup E_i^-} d_e^i \leqslant \sum_{e \in E_i^+ \cup E_i^-} d_e^{i-1}$ **then do**
8. $\quad\quad (G_{i+1}, S_{i+1}, B_i) := \mathbf{GRegion}(G_i, S_i, \mathbf{d}_i, |S_0|, 1)$;
9. $\quad\quad \mu_{i+1}^+ := \max\{0, \mu_i^+ - w(\partial B_i \cap E_i^+)\}$;
10. $\quad\quad \mu_{i+1}^- := \max\{0, \mu_i^- - w(\partial B_i \cap E_i^-)\}$;
11. $\quad\quad i := i + 1$;
12. $\quad$ **Else** next :=false;
13. **For** $j = 0$ **to** $i - 1$ **do**
14. $\quad (H, S, T_j) := \mathbf{GRegion}(G_{j+1}, S_{j+1}, \mathbf{d}_j, |S_0|, +\infty)$;
15. $\quad \mathsf{Cut}_j := \partial B_0 \cup \cdots \cup \partial B_j \cup T_j$;
16. $\quad \Delta E_j := \mathbf{ExtractDeltaE}(\mathsf{Cut}_j)$;
17. $\ell := \arg\min\{w(\Delta E_j) \mid j \in [0, i-1]\}$;
18. **Return** ΔE_ℓ.

Fig. 4 Algorithm **BMulticut**

region growing steps, provided that this is expected to lead to a better solution. Contrast this with the standard region growing algorithm that solves the relaxed integer program only once, before the region growing process starts. Since solving linear programs comes at a cost, however, we limit the number of linear programs to be solved by a parameter K. Solutions to the BOUNDED MULTICUT problem are then obtained from the produced multicut T by a post-processing step. That is, T is split into ΔE such that $T \setminus \Delta E$ is a valid multicut relative to the given bounds μ^+ and μ^-. The set ΔE is returned by the algorithm.

Due to nature of the region growing process, however, it does not necessarily hold that increasing K leads to better solutions. For this reason, for a given K, **BMulticut** runs (at most) K region growing processes, corresponding to the parameter values $1, 2, \ldots, K$ and takes the best solution produced by any of these processes. This clearly guarantees that the quality of solutions never degrades with increasing K.

The remainder of this section is organized as follows. First we detail the **BMulticut** algorithm in Sect. 5.1 and show that it is indeed an approximation algorithm in Sect. 5.2. We conclude with some remarks in Sect. 5.3.

5.1 Algorithm BMulticut

The pseudo-code for the **BMulticut** algorithm is shown in the Fig. 4. It takes as input a graph $G_0 = (V, E_0 = E_0^+ \cup E_0^-, w)$, a set S_0 of source–sink pairs, bounds μ_0^+ and μ_0^-, and the parameter K. In its first step (line 1) it solves the relaxation of IP_{BMC} for G_0, S_0, μ_0^+ and μ_0^-. The relaxation is obtained, as usual, by replacing the constraint (v) in IP_{BMC} (shown in Sect. 4.2) with $x_{uv}, y_{uv} \in [0, 1]$. Note that we can obtain a solution for the relaxation of IP_{BMC} in PTIME by using its equivalent polynomially sized linear program, as discussed in Sect. 4.2. Denote by d_e^0 the returned valuations for variables x_{uv}, where $e = (u, v) \in E_0$. Let $\mathbf{d}_0$ be the collection of all d_e^0, for $e \in E_0$. We ignore the valuations returned for the other variables. In the next step (line 2), the algorithm grows a single region B_0 in G_0, using $\mathbf{d}_0$ by a call to the procedure **GRegion** in which the last parameter is set to 1, indicating that only a single region is requested. As will be detailed below, the region B_0 consists of all vertices that are within a certain distance (relative to $\mathbf{d}_0$) from a source vertex s in S_0. Furthermore, if we denote by ∂B_0 the set of edges in E_0 that have exactly one vertex in B_0, then removing the edges in ∂B_0 from G_0 ensures that the source–sink pair (s, t) in S_0 is disconnected. The procedure **GRegion** returns B_0, the updated graph G_1 obtained from G_0 by removing all vertices (and their incident edges) in B_0, and updated set S_1 of source–sink pairs obtained from S_0 by removing all pairs that have been disconnected by removing ∂B_0 from G_0. Next, **BMulticut** updates the bounds μ_0^+ and μ_0^- (line 3). Intuitively, μ_1^+ measures how much of the bound μ_0^+ is left if all $+$-labeled edges in ∂B_0 are to be part of a valid multicut. Similarly for μ_1^-.

The algorithm **BMulticut** then continues by repeatedly (i) solving a new linear program (line 6); and (ii) growing a single region (line 8) in a similar way as just described. More specifically, the while loop (lines 5–12) is executed as a long as there are still connected source–sink pairs in S_i, no more than K iterations have been performed, when the updated bounds differ from the previous bounds, or when the updated linear program is expected to return a better solution. The latter is checked by the condition in line 7, as will become clear in the analysis of the algorithm.

After completion of the while loop, the value $i - 1$ ($\leqslant K$) denotes the number of times the while loop has been executed. At this point, we have regions $B_0, B_1, \ldots, B_{i-1}$, graphs $G_0 \supsetneq G_1 \supsetneq \cdots \supsetneq G_i$, sets of source–sink pairs $S_0 \supsetneq S_1 \supsetneq \cdots \supsetneq S_i$, and solutions $\mathbf{d}_0, \ldots, \mathbf{d}_{i-1}$ of the (different) linear programs considered.

For each $j \in [0, i-1]$, **BMulticut** will run the standard region growing algorithm (line 14) on G_{j+1} generating a multicut T_j that disconnects all pairs in S_{j+1}. Furthermore, the region growing process uses $\mathbf{d}_j$ to measure the distances between vertices in G_{j+1}. Note that the procedure **GRegion** is called with its last parameter set to ∞, indicating that no upper bound is set on the number of grown regions.

Together with $\partial B_0 \cup \cdots \cup \partial B_j$, the set T_j of edges forms a complete multicut, denoted by Cut_j, for G_0 and S_0 (line 15). It now remains to extract from Cut_j the minimal set of edges ΔE_j such that the remaining edges in Cut_j form a valid multicut relative to μ_0^+ and μ_0^-. This is done by procedure **ExtractDeltaE**(Cut_j) on line 16. This procedure

GRegion $(G, S, \mathbf{d}, \ell, N)$
1. $F := \sum_{e \in E^+ \cup E^-} w_e d_e$;
2. $\epsilon := 2\ln(\ell + 1)$;
3. Initialize $H := G$, $T := \emptyset$; $i = 1$;
4. **While** $(|S| > 0$ **and** $i \leqslant N)$ **do**
5. grow := true
6. pick a source-sink pair (s, t) from S and let region $:= \emptyset$;
7. Let L be the list of vertices in H, sorted by their increasing $\mathbf{d}$-distance to s; Assume that s is the first element $L[0]$ in this list and let $L = L[0]$;
8. **While** (grow) **do**
9. region := region $\cup$ L;
10. update volume $\mathcal{V}$(region) and cost w(region);
11. if c(region) $\leqslant \epsilon \mathcal{V}$(region) then grow := false, else let $L := L$.next;
12. Update H by removing all vertices (and incident edges) in region;
13. Remove all pairs in S that are disconnected in H;
14. $T := T \cup \partial$(region);
15. $i := i + 1$;
16. **Return** (H, S, T).

Fig. 5 Region growing procedure **GRegion**

simply removes edges from Cut_j until $w(\mathsf{Cut}_j \cap E_0^+) \leqslant \mu_0^+$ and $w(\mathsf{Cut}_j \cap E_0^-) \leqslant \mu_0^-$ hold. The removed edges are then returned as ΔE_j. We show below that ΔE_j is indeed a solution of the BOUNDED MULTICUT problem for G_0, S_0 and μ_0^+ and μ_0^-, for each $j \in [0, i-1]$.

Finally, **BMulticut** identifies the solution among the ΔE_j, for $j \in [0, i-1]$, that is of minimal weight (lines 17) and returns it as a solution for the BOUNDED MULTICUT problem (line 18).

We next describe the region growing procedure **GRegion** as shown in Fig. 5. It takes as input a graph $G = (V, E^+ \cup E^-, w)$, set S of source–sink pairs, a collection $\mathbf{d} = \{d_e | e \in E\}$ of values (obtained from solving a linear program, as explained earlier), parameter ℓ which will always be equal to $|S_0|$, and a parameter N that takes values 1 or $+\infty$, depending on whether one, or all regions should be grown. Except for the parameters ℓ and N, **GRegion** is just the standard region growing algorithm [10]. We include it here to make the description of the algorithms self-contained.

Initially, **GRegion** sets $F = \sum_{(u,v) \in E^+ \cup E^-} w_e d_e$ and $\epsilon = 2\ln(\ell + 1)$, where ℓ is the number of source–sink pairs in S_0 (lines 1–2). Then, as long as there are source–sink pairs in S that still need to be disconnected either one (if $N = 1$) or multiple ($N = +\infty$) regions are grown (lines 4–15). A single region region is grown starting from one source vertex at a time. Let (s, t) be the selected source–sink pair (line 6). Next, vertices are added to region in the order determined by their distance to s as given by the values d_e (line 7). Whenever we expand region (line 9), we update its *volume* by setting $\mathcal{V}$(region) $= F/\ell + \sum_{e \cap \text{region} \neq \emptyset} w_e d_e$ (line 10). That is, we add $w_e d_e$ to the volume for every edge that has at least one vertex in region. At the same time, we update its *cost* by setting c(region) $= \sum_{e \in \partial(\text{region})} w_e$, where ∂(region) consists of all edges that have a single vertex in the current region (line 10).

Crucial in the region growing process is to determine when to stop adding vertices to region. For this purpose, the stopping condition c(region) $\leqslant \epsilon \mathcal{V}$(region) is checked (line 11). If the condition holds, no more vertices are added to region by setting grow=false. Otherwise, region is expanded further. It is known that the stopping condition will be satisfied at some point [10]. Furthermore, it is known that after the growing of region is completed, the selected source–sink pair (s, t) becomes disconnected when the edges in ∂(region) are removed from the graph [10].

The algorithm **GRegion** will repeatedly grow regions, disconnecting a source–sink pair at a time, until all the pairs have been processed. This is achieved by updating the graph under consideration (line 12) and updating the set of source–sink pairs such that those that have already been disconnected are disregarded (line 13), after which the algorithm grows a new region in the updated setting.

Furthermore, while growing regions the algorithm collects all edges in ∂(region) (line 14), as this will constitute a multicut T for G and S (when $N = +\infty$) and single region disconnecting (s, t) (when $N = 1$). Finally, observe that the variable i is only to limit the number of regions grown (line 15 together with condition in while loop on line 4). The algorithm returns the updated graph H, updated set S and set of edges T (line 16).

5.2 Correctness and approximation guarantee

We first show that **BMulticut** indeed returns a solution ΔE of the BOUNDED MULTICUT problem on input G_0, S_0, μ_0^+ and μ_0^-. It suffices to show that each ΔE_j, for $j \in [0, i-1]$, is a solution. Recall that ΔE_j is the result of applying **ExtractDeltaE** on Cut_j and that $\mathsf{Cut}_j = \partial B_0 \cup \cdots \cup \partial B_j \cup T_j$. Clearly, if we can argue that Cut_j is a multicut for G_0 and S_0, then ΔE_j is indeed a solution. To see this, recall that **ExtractDeltaE** simply removes edges from Cut_j until it becomes a valid multicut relative to μ_0^+ and μ_0^-, and puts the removed edges in ΔE_j.

It thus remains to show that Cut_j is a multicut for G_0 and S_0. This, however, follows immediately from the correctness of the region growing process [10]. Indeed, it is known that when a region region is grown by **GRegion**, starting from a source vertex s in a pair $(s, t) \in S_0$, then the stopping condition is satisfied by only considering vertices v whose distance to s is strictly less than $\frac{1}{2}$. This implies that the distance between any two vertices in region is strictly smaller than 1. Since the distances are measured using $\mathbf{d}$, a solution of the relaxed integer program IP_{BMC}, the distance between s and t is at least 1 [condition (i) in IP_{BMC}]. In other words, t does

not belong to region and cutting away $\partial(\mathsf{region})$ disconnects s and t. Because **BMulticut** keeps growing regions until no more source–sink pairs are left, it follows that cutting away $\partial B_0 \cup \cdots \cup \partial B_j \cup T_j$ indeed disconnects all source–sink pairs.

Next, we verify that **BMulticut** is an approximation algorithm. We start by bounding the weights of ∂B_j and T_j for $j \in [0, i-1]$ in terms of $F_0 = \sum_{e \in E_0^+ \cup E_0^-} w_e d_e^0$. Consider the region B_j. We know from the stopping condition used in **GRegion** that $c(B_j) \leqslant 2\ln(|S_0|+1)\mathcal{V}(B_j)$ where the cost and volume are computed using $\mathbf{d}_j$. Since $c(B_j) = w(\partial B_j)$ we thus have that

$$\begin{aligned} w(\partial B_0 \cup \cdots \cup \partial B_j) &= \sum_{i=0}^{j} c(B_i) \\ &\leqslant \sum_{i=0}^{j} 2\ln(|S_0|+1)\mathcal{V}(B_i). \end{aligned}$$

Similarly, $w(T_j) \leqslant 2\ln(|S_0|+1)\sum_{\mathsf{region}^j} \mathcal{V}(\mathsf{region}^j)$, where the summation runs over all (say p_j) regions constructed to obtain T_j. For $j \in [0, i-1]$ denote by $F_j = \sum_{e \in E_j^+ \cup E_j^-} w_e d_e^j$ where d_e^j denote the values in $\mathbf{d}_j$ and E_j^+ and E_j^- are the edges in G_j. Then,

$$\mathcal{V}(B_j) = F_j/|S_0| + \sum_{e \cap B_j \neq \emptyset} w_e d_e^j$$

and

$$\sum_{\mathsf{region}^j} \mathcal{V}(\mathsf{region}^j) = p_j \frac{F_j}{|S_0|} + \sum_{e \in E_{j+1}^+ \cup E_{j+1}^-} w_e d_e^j.$$

This implies that $w(\mathsf{Cut}_j)$ is bounded by

$$\begin{aligned} 2\ln(|S_0|+1)\Bigg(&\frac{1}{|S_0|}(F_0 + F_1 + \cdots + F_{j-1} + (p_j+1)F_j) \\ &+ \Bigg(\sum_{e \cap B_0 \neq \emptyset} w_e d_e^0 + \sum_{e \cap B_1 \neq \emptyset} w_e d_e^1 + \cdots \\ &\quad + \sum_{e \cap B_j \neq \emptyset} w_e d_e^j + \sum_{e \in E_{j+1}^+ \cup E_{j+1}^-} w_e d_e^j\Bigg)\Bigg). \end{aligned}$$

Note that **BMulticut** ensures (by the condition in line 7) that

$$\begin{aligned} F_j = \sum_{e \in E_j^+ \cup E_j^-} w_e d_e^j &\leqslant \sum_{e \in E_j^+ \cup E_j^-} w_e d_e^{j-1} \\ &\leqslant F_{j-1} = \sum_{e \in E_{j-1}^+ \cup E_{j-1}^-} w_e d_e^{j-1}, \end{aligned} \quad (1)$$

for any $j \in [1, i-1]$. Furthermore, $j + p_j + 1$ is at most $|S_0|$ since, in the worst case, $|S_0|$ regions are grown each of which disconnects a single source–sink pair in S_0. Hence, $\frac{1}{|S_0|}(F_0 + F_1 + \cdots + F_{j-1} + (p_j+1)F_j) \leqslant \frac{1}{|S_0|}|S_0|F_0 = F_0$. Similarly,

$$\begin{aligned} F_0 &= \sum_{e \cap B_0 \neq \emptyset} w_e d_e^0 + \sum_{e \in E_1^+ \cup E_1^-} w_e d_e^0 \\ &\geqslant \sum_{e \cap B_0 \neq \emptyset} w_e d_e^0 + F_1 \\ &= \sum_{e \cap B_0 \neq \emptyset} w_e d_e^0 + \sum_{e \cap B_1 \neq \emptyset} w_e d_e^1 + \sum_{e \in E_2^+ \cup E_2^-} w_e d_e^1 \\ &\geqslant \sum_{e \cap B_0 \neq \emptyset} w_e d_e^0 + \sum_{e \cap B_1 \neq \emptyset} w_e d_e^1 + F_2 \\ &\vdots \\ &\geqslant \sum_{e \cap B_0 \neq \emptyset} w_e d_e^0 + \sum_{e \cap B_1 \neq \emptyset} w_e d_e^1 + \cdots \\ &\qquad + \sum_{e \cap B_j \neq \emptyset} w_e d_e^j + \sum_{e \in E_{j+1}^+ \cup E_{j+1}^-} w_e d_e^j. \end{aligned}$$

All combined this implies that $w(\mathsf{Cut}_j) \leqslant 4\ln(|S_0|+1)F_0$. Note, however, that we are interested in approximating solution of the Bounded Multicut problem. Let $\mathbf{d}_0$ be a valuation of the variables x_{uv} and $\mathbf{f}_0$ be a valuation of the variables y_{uv} in the relaxation of $\mathrm{IP}_{\mathrm{BMC}}$. Then,

$$\left(\sum_{e \in E_0^+ \cup E_0^-} w_e d_e^0\right) - \left(\mu_0^+ + \mu_0^-\right) \leqslant \sum_{e \in E_0^+ \cup E_0^-} w_e \left(d_e^0 - f_e^0\right)$$

and thus

$$F_0 \leqslant |\Delta E_{\mathrm{lp}}| + \left(\mu_0^+ + \mu_0^-\right),$$

where $|\Delta E_{\mathrm{lp}}|$ denotes the objective value of the relaxation of $\mathrm{IP}_{\mathrm{BMC}}$. Hence,

$$\begin{aligned} w\left(\Delta E_j\right) \leqslant w\left(\mathsf{Cut}_j\right) &\leqslant 4\ln\left(|S_0|+1\right)F_0 \\ &\leqslant 4\ln\left(|S_0|+1\right)\left(|\Delta E_{\mathrm{opt}}| + \left(\mu_0^+ + \mu_0^-\right)\right) \end{aligned} \quad (2)$$

since $|\Delta E_{\mathrm{lp}}| \leqslant |\Delta E_{\mathrm{opt}}|$ where $|\Delta E_{\mathrm{opt}}|$ is the size of the optimal solution as given by the integer program $\mathrm{IP}_{\mathrm{BMC}}$.

We thus have indeed obtained a $O(\log|S_0|)$-approximation algorithm for Bounded Multicut. To relate this back to the Bounded Correlation problem, observe that S_0 corresponds to the negative edges (recall that the transformation from Bounded Correlation Clustering to Bounded Multicut), and thus **BMulticut** is a $O(\log(|E^-|))$-approximation algorithm for Bounded Correlation Clustering.

5.3 Remarks

Observe that when $K = 0$, the algorithm **BMulticut** is precisely the same as the standard region growing algorithm. It is thus a conservative extension of the standard algorithm. Furthermore, the analysis of **BMulticut** reveals that we expect that increasing K leads to better solutions. Indeed, the inequality (1) in the analysis shows that $w(\mathsf{Cut}_j)$ can be bounded in terms of the sum of the F_j's rather than F_0 alone, as is the case when setting $K = 0$. Since $F_j \leqslant F_{j-1}$ for $j \in [1, i-1]$ and thus $F_j \leqslant F_0$ for $j \in [0, i-1]$, this means that the weight of the multicut (and thus also ΔE) can be bounded by a smaller number than $4\ln(|S_0|+1)F_0$. Of course, this does not imply that we always get a better solution since these are upper bounds. This is precisely the reason why **BMulticut** takes the minimum solution among all those that are generated.

Furthermore, the region growing algorithm uses the relaxation of IP_{BMC} to determine, among other things, the order in which vertices are added to the region. However, the algorithm only uses the valuations d_{uv} for the variables x_{uv} and does not explicitly leverages the availability of valuations f_{uv} for y_{uv}. As an immediate consequence, the weight of the obtained multicut is related to $F = \sum_{(u,v)\in E^+\cup E^-} w_{uv}d_{uv}$ rather than the objective function of IP_{BMC} ($\sum_{(u,v)\in E^+\cup E^-} w_{uv}(d_{uv}-f_{uv})$). We thus get an overly pessimistic upper bound on the quality of the approximation. Nevertheless, we will see in the experimental section that we get ΔE's that are close to optimal.

Ideally, we would like to grow regions using $d_{uv} - f_{uv}$ rather than d_{uv}. This does not work, however, since constraint (i) in IP_{BMC} only refers to the x_{uv} variables and this constraint, together with the stop condition for growing regions (line 11 in the algorithm **GRegion**), ensures that the result is indeed a multicut. Further investigation is required as how to make better use of the available valuations of the y_{uv} variables.

However, we remark that the additional constraints (ii), (iii), and (iv) in IP_{BMC} may indeed help to guide the region growing process toward a better solution than when ignoring these constraints and using the relaxation of IP_{MC} for the standard MULTICUT problem as is illustrated in the following example.

Example 5 Consider a simple instance of BOUNDED MULTICUT for $\mu^+ = 1$ and $\mu^- = 1$ as shown in Fig. 6, i.e., we allow for one positive and one negative edge to belong to the multicut. As before, dashed edges represent edges in E^-, solid edges correspond to edges in E^+. We adorned the edges with valuations for x_{uv} obtained by relaxing IP_{MC} (left) and with valuations for x_{uv} and y_{uv} obtained by relaxing IP_{BMC} (right). In this example, we get integer solutions. As can be seen, by growing regions based on IP_{MC}, we get a multicut consisting of two positive edges (the regions are gray-shaded). After post-processing, we put one of these in ΔE and thus $|\Delta E| = 1$. However, by growing regions based on IP_{BMC}, we immediately obtain a multicut that is valid, i.e., it consists of one positive and one negative edge. As a consequence, an empty ΔE will be returned by the algorithm. □

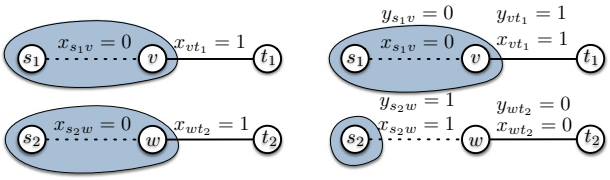


Fig. 6 Effect of additional constraints on approximate solution

The example shows the advantage of growing regions based on the relaxation of IP_{BMC}.

6 Interactive framework

Before showing our experimental evaluation of the **BMulticut** algorithm, we describe how it fits in the user interaction process illustrated in Fig. 1. Recall from that example that a user can (1) specify false negative and positive bounds; (2) mark a set of edges as immutable and make immutable edges mutable again; and (3) receive feedback on revised bounds suggested by the system when no solution can be found for the given error bounds and set of immutable edges.

Clearly, the error bounds are well-integrated in **BMulticut** since they are part of the linear program that is solved to obtain a solution, as explained in the previous section. We therefore focus on (2) and (3). We discuss these issues in the context of BOUNDED MULTICUT but as we have seen before, this discussion also concerns BOUNDED CORRELATION CLUSTERING.

6.1 Processing of immutable edges

When a user declares an edge as immutable, she/he does not want any solution ΔE that includes this edge. Recall the integer program IP_{BMC} for solving the BOUNDED MULTICUT problem. Note that according to semantics of a valuation of variables x_{uv} and y_{uv}, an edge $e = (u, v)$ is in ΔE (and hence to be deleted) if $x_{uv} = 1$ and $y_{uv} = 0$. To prevent e to be in ΔE it thus suffices to set $x_{uv} = 0$ in IP_{BMC}.

Therefore, given a set of immutable edges IE from a user during some stage in the interaction process, we proceed as follows: for every edge $e = (u, v) \in IE$, we add an equality constraint of the form: $x_{uv} = 0$ to the IP_{BMC} program. Note that we add at most $|IE|$ constraints and thus the program remains polynomial in size.

The algorithm **BMulticut** then uses the relaxation of this modified integer program. When growing regions, any valuation of these x_{uv} will be equal to 0 and hence the corresponding edges will always be included in a region region and not in its boundary ∂(region). Indeed, **GRegion** adds vertices to region in increasing distance from a source vertex, hence those with distance zero will always be added to region. In other words, immutable edges will never be part of ∂(region) and thus also not be part of Cut. As a consequence, edges in IE will not be present in a solution ΔE. Indeed, recall that ΔE is obtained as a subset of Cut.

6.2 Obtaining revised error bounds

Although a solution always exists in the absence of immutable edges, this is no longer the case when immutable edges are present as was shown in Example 1. Intuitively, in these cases the integer program becomes over-constrained and no solution exists. In particular, when too many variables x_{uv} are set to zero, it may become impossible to satisfy condition (i) in IP_{BMC}. In such cases, as part of the user interaction we want to suggest revised error bounds that guarantee the existence of a solution.

To deal with this problem observe that IP_{BMC} becomes infeasible when there are source–sink pairs that have a path between them that entirely consists of immutable edges, i.e., edges (u, v) whose x_{uv}-value is 0. Moreover, such source–sink pairs can be easily found in PTIME by starting from a source vertex and checking whether the corresponding sink vertex can be reached by following immutable edges only. Intuitively, such an immutable path cannot be disconnected by removing edges as all edges are immutable and should not be deleted, as requested by the user.

Given a set S of source–sink pairs, we denote by S_{ie} those pairs for which an immutable path exists, and by S_{nie} the remaining source–sink pairs in S. During the interactive process, we maintain the sets S_{ie} and S_{nie} and modify **BMulticut** as follows:

- Instead of using the relaxation of IP_{BMC}, we use the relaxation of the revised integer program obtained by including $x_{uv} = 0$ for each $(u, v) \in IE$ and by only considering paths between source–sink pairs in S_{nie} in condition (i); This guarantees the feasibility of the revised integer program and its relaxation.
- When applying **BMulticut** with this revised relaxation, we obtain a solution ΔE that does not contain any immutable edges and for which there exists a valid multicut T. In other words, removing the edges $\Delta E \cup T$ disconnects all source–sink pairs in S_{nie}.
- We further expand T with a multicut for the source–sink pairs in S_{ie}. To this aim, we run **BMulticut** in which the error bounds are set to 0 and for which only source–sink pairs in S_{ie} are taking into account. Furthermore, in the corresponding integer program, we do treat immutable edges as normal edges by removing the additional equality conditions $x_{uv} = 0$ for $(u, v) \in IE$. By setting the error bounds to zero, **BMulticut** generates a T' whose removal disconnects all pairs in S_{ie}. In other words, $T \cup T'$ is a multicut for the original set S albeit for error bounds $(\mu^+)' = \mu^+ + w(T' \cap E^+)$ and $(\mu^-)' = \mu^- + w(T' \cap E^-)$.

That is, when a user marked too many edges as immutable such that no solution can be found for the given error bounds μ^+ and μ^-, the user is informed about this and in addition, she/he is informed that a solution exists for revised upper bounds $(\mu^+)'$ and $(\mu^-)'$. If these revised bounds are acceptable for the user, the interactive process can then continue. Otherwise, a user may decide to make certain immutable edges mutable (normal) again.

Example 6 Consider the triangle $(1, 4)$, $(1, 2)$ and $(2, 4)$ from Example 1 that was marked as immutable. Recall that $S = \{(z_{1,4}, 4), (z_{3,6}, 6), (z_{5,6}, 6)\}$ in the corresponding BOUNDED MULTICUT instance shown in Fig. 2. Note that the pair $(z_{1,4}, 4)$ has an immutable path: $(z_{1,4}, 1)$, $(1, 2)$, $(2, 4)$ where the edge $(z_{1,4}, 1)$ corresponds to the $-$-labeled edge $(1, 4)$. Hence, $S_{ie} = \{(z_{1,4}, 4)\}$ and $S_{nie} = \{(z_{3,6}, 6), (z_{5,6}, 6)\}$. We have seen in Example 1 that no solution exists for $\mu^+ = 0$ and $\mu^- = 0$. Indeed, to disconnect $(z_{1,4}, 4)$ one of the edges $(z_{1,4}, 1)$, $(1, 2)$, $(2, 4)$ needs to be deleted, which is disallowed since these are marked as immutable. Following the strategy outline above, we thus first process source–sink pairs in S_{nie}, leading to a solution $\Delta E = \{(3, z_{3,6}), (5, z_{5,6})\}$ that disconnects all pairs in S_{nie}, followed by a cut $T' = \{(z_{1,4}, 1)\}$ disconnecting the pair in S_{ie}. Hence $(\mu^+)' = \mu^+ = 0$ and $(\mu^-)' = \mu^- + w(T' \cap E^-) = 0 + 1 = 1$. Hence, $(\mu^+)' = 0$ and $(\mu^-)' = 1$ will be suggested to the user as new error bounds for which a solution exists. The corresponding clustering $\mathcal{C}$ and solution ΔE are shown in Fig. 1g. □

7 Experimental evaluation

We now describe the empirical evaluation of our approximation algorithm on synthetic and real datasets. For solving the integer program IP_{BMC} and its relaxation, we use the IBM Cplex Optimizer [14]. The algorithm **BMulticut** is implemented in Java. The experiments were conducted on a GNU/Linux machine with Intel(R) Xeon(R) CPU 2.90 GHz (16 cores) and 32 GB memory.

7.1 Datasets

We perform experiments on synthetic and real datasets, which are detailed next.

7.1.1 Synthetic data

We use two kinds of synthetic data. The first kind of synthetic data synth_cc is aimed particularly at correlation clustering. Here, the graph data are generated along the same lines as in [4]. That is, starting with an initial number of vertices and number of clusters, we randomly add "+" and "-" labeled edges to the graph using parameters to control the number of false positive and negatives. These parameters include the probability of intra-cluster edges, probability of inter-cluster edges and the probability of having a negative edge inside a cluster. The second kind of synthetic data synth_gen concerns randomly generated graphs in which the number of clusters and errors is purely random. That is we do not control number of clusters or number of false negatives and positives. However, we do control the total number of "-" labeled edges in the graph.

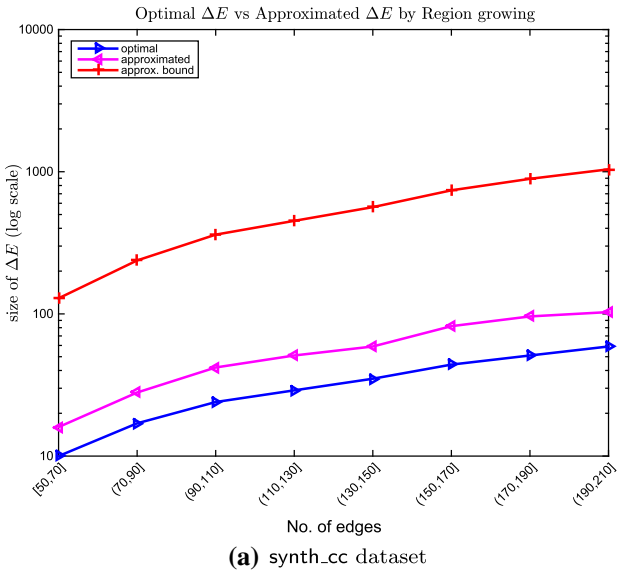
7.1.2 Real data

We used two real datasets: (1) The epinions social network dataset [15] and (2) the Wikipedia requests for adminship dataset [19].

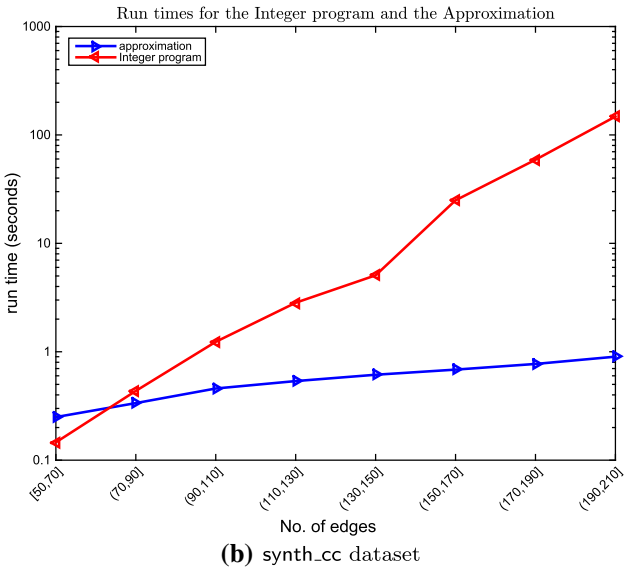
The epinions social network dataset is a directed graph depicting a who-trust-whom network from a general consumer site epinions.com. The directed edges in this dataset come with a $+$ or $-$-label indicating who-trust-whom. To turn this into an undirected graph, we proceeded as follows. For those pairs of vertices that only had a single directed edge between them, we removed the direction. For those pairs that had edges in both directions, we replaced them with a single undirected edge and randomly picked a label if the original two directed edges carried distinct labels. We sampled 10 subgraphs with sizes ranging from 1647 to 7808 in the number of vertices and 1000 to 5500 in the number of edges.

The Wikipedia requests for adminship dataset is a signed network in which nodes represent Wikipedia users and the edges represent votes. In order for a Wikipedia editor to become an administrator, the candidate or another member must submit a Request for Adminship (RfA). Thereafter, any community member may cast a vote indicating their support or opposition. A vote may also be neutral. This directly translated to our problem instance in which a supporting vote becomes a "+"-labeled edge and an opposing edge becomes a "-"-labeled edge. We ignored the neutral votes. Similarly, subgraphs were sampled from the full dataset ranging in size from 1847 to 2367 in the number of vertices and 1500 to 2250 in the number of edges. We varied the false negative and positive bounds between 0 and 25 for both datasets.

In all experiments, we report averaged results over 20 runs.



(a) synth_cc dataset



(b) synth_cc dataset

Fig. 7 Quality of approximation and running times

7.2 Quality of approximation

We first investigate the quality of solutions ΔE returned by **BMulticut** by comparing it against the optimal solution ΔE_{opt}, obtained by solving the integer program IP_{BMC}. For the synth_cc dataset, we considered small graphs ranging from 50 to 210 in number of edges. A total of 640 graph instances were generated. We considered small graphs in this part of the experiment because as we see in Fig. 7a, even for small graphs, a clear behavior of the algorithm in terms of the approximate, optimal solutions and the theoretical bound is observed. This behavior is expected to hold for larger graphs as well.

Figure 7a shows the size of the optimal ΔE_{opt}, the size of set of ΔE returned by **BMulticut**, and the theoretical

upper bound on the approximation guarantee, in terms of the number of edges in the input graph and error bounds. The theoretical bound corresponds to Eq. (2) in Sect. 5.2. Each of the points shown in the graph represents averaged results over 80 input instances of the synth_cc data sets. The parameter K and both error bounds μ_{fp} and μ_{fn} were set to zero. Since the quality of solutions returned by **BMulticut** does not degrade for increasing values of parameter K, we can only expect to get solutions that lie closer to the optimal solution for $K > 0$. One can see from Fig. 7a that **BMulticut** obtains solutions that are consistently close to the optimal and much better than predicted by the theoretical approximation guarantee. This shows that **BMulticut** generates good solutions. A similar behavior (not shown) was observed on the synth_gen data sets.

We then checked the effect of increasing the error bounds on the size of ΔE returned by **BMulticut**. Also here the parameter K is set to zero. Figure 8a, b show the size of ΔE returned by **BMulticut** for varying error bounds on the synth_cc and Wikipedia datasets, respectively. For Fig. 8a, reported results are for individual graph instances selected from the synth_cc data sets. Not surprisingly, as we increase the error bounds the size of ΔE decreases. Indeed, one expects that less edges need to be deleted to guarantee the existence of clusterings with large cost (high error bounds).

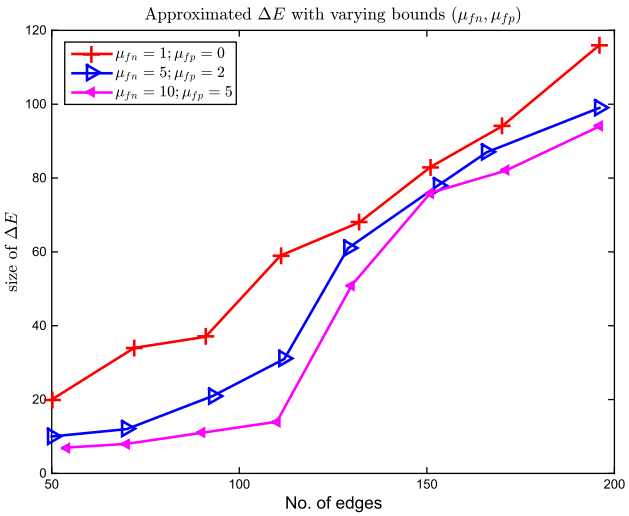
Note that for the Wikipedia dataset results are reported for graphs starting from 1500 edges. The reason is that for graphs of less than 1500 edges with parameters K and error bounds fixed to zero, the approximated ΔE matched the optimal solution ΔE_{opt}, and more specifically $\Delta E = \emptyset$. For this reason we do not show the results for these smaller samples of the Wikipedia dataset. The results on the epinions dataset were similar (not shown).

The plots shown in Fig. 8a, b illustrate ranges of ΔE that are achievable with different sizes of error bounds. Since the edges in ΔE are returned for user inspection, we want the size of ΔE to be reasonable. To this aim, we explore in a bit more details how large the ΔE's returned by **BMulticut** are, compared to the total number of edges in the graph. We report our findings in Table 1 for a couple of synth_cc data sets. As can be seen, only a small fraction of edges are returned for user inspection.

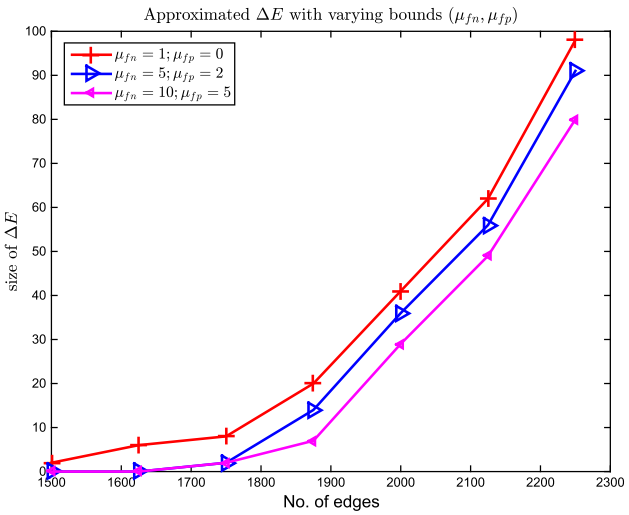
For the real datasets, the returned ΔE were generally very small. For instance, in the epinions dataset, for the sampled graphs an average ΔE of size 6 was obtained, translating into an average ratio ($|\Delta E|/|E^+ \cup E^-|$) of 0.002. Similarly, for the Wikipedia dataset, we obtained an average ratio of 0.005. As before, in all these experiments K was set to zero.

7.3 Scalability

We next compare the running times of solving the integer program IP_{BMC}, and thus obtaining an exact solution, against



(a) synth_cc dataset



(b) Wikipedia dataset

Fig. 8 Effect of bounds

Table 1 synth_cc dataset: ratio of $|\Delta E|$ versus $|E^+ \cup E^-|$

No. vertices $\lvert V\rvert$	No. edges $\lvert E^+ \cup E^-\rvert$	$\lvert\Delta E\rvert/\lvert E^+ \cup E^-\rvert$
250	634	0.23
270	1250	0.13
290	2094	0.07
300	2566	0.06
310	3119	0.06
330	4389	0.04
450	5739	0.05

the running time of **BMulticut**. Figure 7b shows the averaged running times for 80 instances of synth_cc. We varied the error bounds between 0 and 10 with parameter K fixed at zero. Not surprisingly, solving IP_{BMC} quickly becomes more time consuming as input graphs get larger. Instead, the

Table 2 synth_cc dataset: linear program size

No. vertices $\|V\|$	No. edges $\|E^+ \cup E^-\|$	No. variables	No. constants	No. nonzeros
28	50	564	2029	4840
46	70	1752	6476	15,746
49	90	1830	7532	18,720
56	110	2624	11,575	29,212
66	130	3490	15,627	39,590
88	150	6496	27,115	67,980
92	170	7120	31,502	79,850
103	190	9000	40,052	101,690
115	210	11,305	50,248	127,617

Table 3 synth_cc dataset: LP and region growing run times

No. vertices $\|V\|$	No. edges $\|E^+ \cup E^-\|$	LP runtime (s)	Region growing runtime (s)
250	634	15.56	0.57
270	1250	89.93	0.45
290	2094	189.69	0.89
300	2566	172.18	0.99
310	3119	153.32	1.04
330	4389	196.48	1.51
450	5739	1269.41	2.28

approximation algorithm returns results within reasonable time. For instance, for small graphs in the range $(90, 210]$ in number of edges, the IP_{BMC} program is at least 2 orders of magnitude slower than our approximation algorithm. This scalability behavior is expected to hold when larger graph instances are considered.

We remark that for very small graphs, solving IP_{BMC} is sometimes faster than running the approximation algorithm. Note, however, that these differences are very small (log scale) and probably due to background processes.

Figure 7b shows that the running time of **BMulticut** grows with increasing graph sizes. This is mainly due to the increasing size of the linear programs that need be solved. Table 2 illustrates how the linear program changes in terms of the number of variables, constants and non-zero values (i.e., number of non-zero values in the Cplex matrix representation of the linear program).

Table 3 shows that solving the linear program (LP Runtime in seconds) constitutes the dominant factor in the whole region growing process. In this experiment we set parameter K and both error bounds to zero. Since for this experiment we do not need to solve the program IP_{BMC}, which takes longer, we used larger graph instances of the synth_cc data sets and we show results for a couple of instances in Table 3.

7.4 Effect of the parameter K

So far, we have set the parameter K to zero such that **BMulticut** generates "worst-case" solutions ΔE. Indeed, as observed earlier, increasing the value of the parameter K should lead to better approximate solutions. We next validate this claim experimentally.

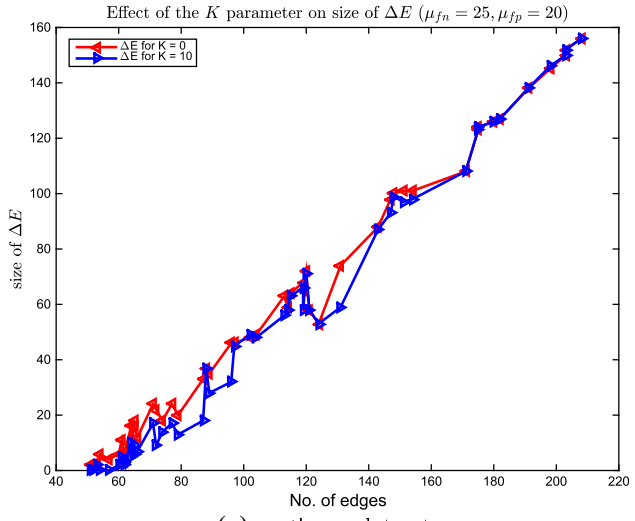
Firstly, for fixed error bounds, we investigate how the sizes of the returned ΔE is affected by the value of K. We therefore compared results for the values of $K = 0$ and $K = 10$. The plots in Fig. 9a, b show results on synth_gen and synth_cc data sets, respectively. Error bounds are set to $\mu_{fp} := 25$ and $\mu_{fn} := 20$.

Figure 10a, b show the results for the Wikipedia dataset. We considered two pairs of error bounds i.e., ($\mu_{fp} = 10$; $\mu_{fn} = 5$) and ($\mu_{fp} = 25$; $\mu_{fn} = 20$), respectively.

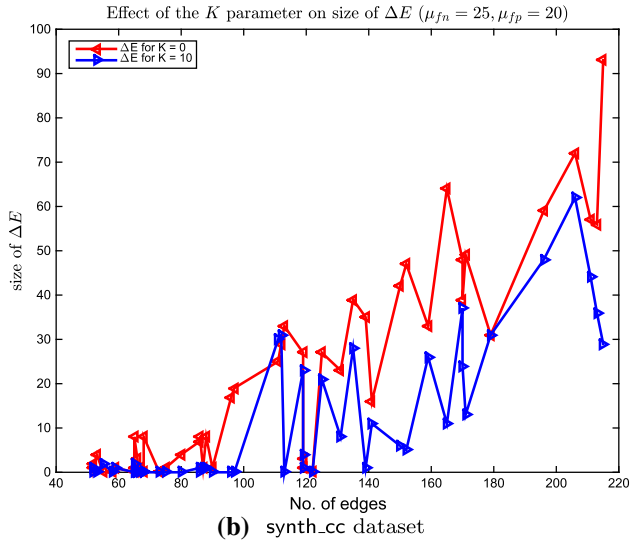
From all these plots, we observe that as K is relaxed from 0 to 10, we indeed get a smaller ΔE in most of the cases. Only in a few instances, the size of ΔE obtained matched the size of ΔE obtained in the zero-valued K case. We may conclude that relaxing K helps us to obtain better solutions than when fixing K to 0.

We also observe that Fig. 9a, b show that the benefit of non-zero values for parameter K in obtaining better solutions is more noticeable for the synth_cc datasets (Fig. 9b) than for the synth_gen datasets (Fig. 9a). Since the synth_cc datasets are generated with some underlying structure mimicking correlation clustering real-life scenarios, our experiments indicate that our algorithm works better in those problem instances (i.e., like synth_cc) than when one has completely random graph instances (i.e., synth_gen datasets).

Since K indicates how many region growing processes **BMulticut** can run, and from which the best solution can

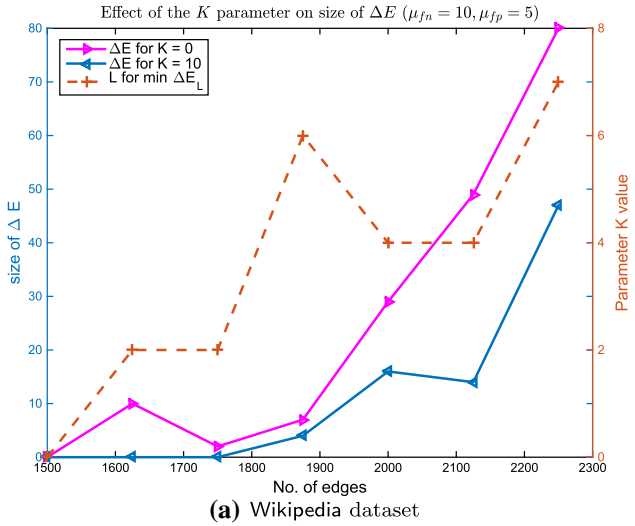


(a) synth_gen dataset

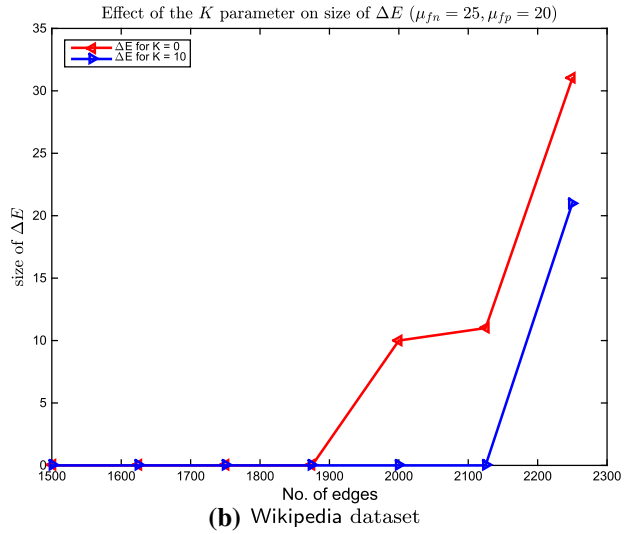


(b) synth_cc dataset

Fig. 9 Effect of the K parameter on ΔE



(a) Wikipedia dataset



(b) Wikipedia dataset

Fig. 10 Effect of the K parameter on ΔE

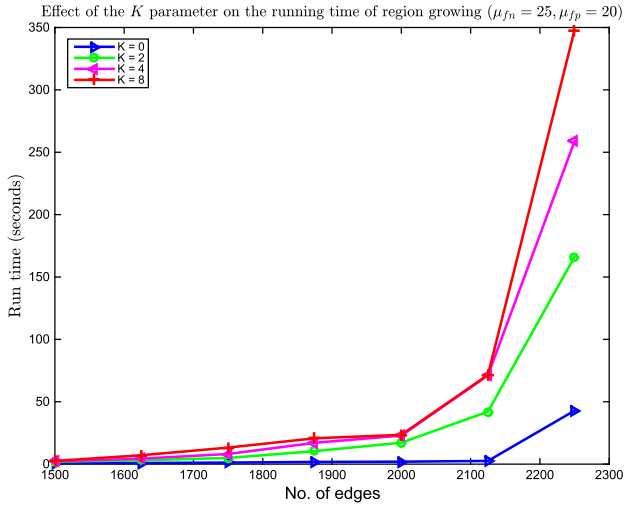
be taken, we also investigated which of these runs actually generated the best solution. In Fig. 10a we show which run actually produced the best solution (dashed line) on the Wikipedia dataset. We observe that in most cases the best solution was obtained by running at most 7 region growing processes. In some cases, even running two region growing processes suffices to obtain the best solution. Only in one case (1500 edges), the best solution was obtained for $K = 0$.

To see the impact of the error bounds on the size of ΔE when K can be larger than 0, consider Fig. 10a, b. As previously observed (for $K = 0$ in Fig. 8a, b), when error bounds are increased we obtain smaller solutions. Indeed, Fig. 10b shows that we obtain smaller sizes of ΔE as compared to the sizes reported in Fig. 10a. Recall that ($\mu_{fp} = 10$; $\mu_{fn} = 5$) and ($\mu_{fp} = 25$; $\mu_{fn} = 20$) in Fig. 10a, b, respectively.

Specifically, looking at the largest instance of 2250 edges and for the $K = 0$ case, as the error bounds are relaxed (i.e., μ_{fp} : 10 to 25 and μ_{fp} : 5 to 20), we note an improvement (decrease) in size of ΔE by a factor of about 2.6. Similarly, when K is set to 10, sizes of ΔE decrease by a factor of approximately 2.2, when error bounds increase.

Furthermore, one would expect the positive effect of larger K to be less prominent when large error bounds are present. Indeed, larger error bounds alone lead to smaller solutions already. Indeed, consider again the instance of 2250 edges in Fig. 10a, b. The difference in the sizes of ΔE between the two cases (i.e., $K = 0$ vs. $K = 10$) in Fig. 10a is 33, while in Fig. 10b the difference is only 10. A similar behavior was noted for the epinions dataset (not shown).

Finally, we report the effect of K on the efficiency of our approximation algorithm. Recall that K corresponds to the number of linear programs that are solved. Hence, increasing K is expected to add an overhead in terms of the running time. Figure 11a, b shows the running times for various K on the



(a) Wikipedia dataset

(b) synth_cc dataset

Fig. 11 Effect of the K parameter on running time of region growing

Wikipedia and synth_cc datasets, respectively. As expected, the smallest running times are obtained when $K = 0$ (i.e., for the Wikipedia dataset the largest instance in this case 2250 edges completed in under 43 s). Similarly, for small synthetic datasets, running times are below 0.8 s when $K = 0$. For the $K > 0$ case, in Fig. 11a we compare the running times for values of K set to 2, 4 and 8. We note that our algorithm remains efficient as K is increased and as before notable differences in time are observed for instances of at least 2000 edges. Moreover, as K was increased from 2 to 8, the maximum increase in time for the largest instance was by a factor of 2.1. Furthermore, we note that there is a slightly higher increase in running time between the 2125-edge and 2250-edge instances. This is attributed to sampling which resulted in graphs which are different in structure. This behavior was similar for the synth_cc datasets as shown in Fig. 11b where K was set to 4, 12 and 20. Observe that for these smaller datasets, even as K as set to 20, **BMulticut** algorithm completed running in under 16 s.

In summary, the running times increase in terms of K, as expected. However, we note that for small values of K, the overheads are smaller for successive values. As we reported earlier, even when K is large, the best solution is sometimes found by running less (than K) region growing processes. In practice, a user may thus adopt a pay-as-you-go strategy in which **BMulticut** is halted whenever a time limit is exceeded. The best solution found in the allowed time is then returned.

8 Related work

Most relevant to our work is the $O(\log n)$-approximation algorithm for CORRELATION CLUSTERING presented in [8]. In that work, the region growing algorithm for MULTICUT [10] is used to obtain an approximation algorithm for CORRELATION CLUSTERING. We follow a similar strategy in this paper, albeit in the presence of error bounds, as explained in Sect. 5.

The CORRELATION CLUSTERING problem itself was introduced in [2]. Approximation algorithms have been reported in [2,6,9,16] with the goal of minimizing disagreements (i.e., minimizing false positive and negatives, which corresponds to our setting) or maximizing agreements. Different algorithms are provided depending on whether graphs are complete or incomplete, weighted or unweighted, or whether certain conditions on the weights are imposed. None of these papers consider error bounds and to our knowledge, the BOUNDED CORRELATION CLUSTERING problem has not been studied before.

Similarly, the MULTICUT problem has received ample attention, see e.g., [7] for a survey. To our knowledge, the BOUDED MULTICUT problem has not been considered before.

We also note that a number of variations of the CORRELATION CLUSTERING problem have been considered: by fixing the number of clusters [12]; by allowing overlapping clusters [5]; and for generally labeled edges (chromatic clustering) [4]. None of these works consider the presence of user-defined error bounds, however.

This work is an extended version of [11]. We refer to Sect. 1 for a description of the differences between the current paper and [11].

9 Conclusion

We revisited CORRELATION CLUSTERING when users can specify desired error bounds. An approximation algorithm is provided for the associated BOUNDED CORRELATION

CLUSTERING problem. We analyzed the algorithm theoretically and provide an experimental validation. It is shown that approximation algorithm provides solutions that are close to optimal. Furthermore, we envisage our algorithm to be part of an interactive clustering framework. We leave the full exploration and implementation of the interactive framework as part of future work. Another direction is to reconsider the variations of the CORRELATION CLUSTERING PROBLEM [4,5,12] mentioned before, in the presence of error bounds.

References

1. Aggarwal, C.C., Reddy, C.K.: Data Clustering: Algorithms and Applications. CRC Press, Boca Raton (2013)
2. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Mach. Learn. **56**(1–3), 89–113 (2004)
3. Bejerano, Y., Smith, M.A., Naor, J.S., Immorlica, N.: Efficient location area planning for personal communication systems. IEEE/ACM Trans. Netw. (TON) **14**(2), 438–450 (2006)
4. Bonchi, F., Gionis, A., Gullo, F., Ukkonen, A.: Chromatic correlation clustering. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1321–1329 (2012)
5. Bonchi, F., Gionis, A., Ukkonen, A.: Overlapping correlation clustering. In: Proceedings of the 11th IEEE International Conference on Data Mining, pp. 51–60 (2011)
6. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. In: Proceedings of the 44th IEEE Symposium on Foundations of Computer Science, pp. 524–533 (2003)
7. Costa, M.-C., Létocart, L., Roupin, F.: Minimal multicut and maximal integer multiflow: a survey. Eur. J. Oper. Res. **162**(1), 55–69 (2005)
8. Demaine, E.D., Emanuel, D., Fiat, A., Immorlica, N.: Correlation clustering in general weighted graphs. Theor. Comput. Sci. **361**(2), 172–187 (2006)
9. Demaine, E.D., Immorlica, N.: Correlation clustering with partial information. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pp. 1–13. Springer, Berlin (2003)
10. Garg, N., Vazirani, V.V., Yannakakis, M.: Approximate max-flow min-(multi) cut theorems and their applications. SIAM J. Comput. **25**(2), 235–251 (1996)
11. Geerts, F., Ndindi, R.: Interactive correlation clustering. In: Proceedings of the 1st International Conference on Data Science and Advanced Analytics, pp. 170–176 (2014)
12. Giotis, I., Guruswami, V.: Correlation clustering with a fixed number of clusters. In: Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithm, pp. 1167–1176 (2006)
13. Hu, T.C.: Multi-commodity network flows. Oper. Res. **11**(3), 344–360 (1963)
14. IBM. Cplex Optimzer. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. Accessed 27 Nov 2013
15. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Signed networks in social media. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1361–1370 (2010)
16. Swamy, C.: Correlation clustering: maximizing agreements via semidefinite programming. In: Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms, pp. 526–527 (2004)
17. Van Gael, J., Zhu, X.: Correlation clustering for crosslingual link detection. In: IJCAI, pp. 1744–1749 (2007)
18. Vazirani, V.V.: Approximation Algorithms. Springer, Berlin (2001)
19. West, R., Paskov, H.S., Leskovec, J., Potts, C.: Exploiting Social Network Structure for Person-to-Person Sentiment Analysis. arXiv preprint arXiv:1409.2450 (2014)