CrossMark

A two-level graph partitioning problem arising in mobile wireless communications

Jamie Fairbrother[1] · **Adam N. Letchford[2]** ·
Keith Briggs[3]

Abstract In the *k-partition problem* (*k*-PP), one is given an edge-weighted undirected graph, and one must partition the node set into at most k subsets, in order to minimise (or maximise) the total weight of the edges that have their end-nodes in the same subset. Various hierarchical variants of this problem have been studied in the context of data mining. We consider a 'two-level' variant that arises in mobile wireless communications. We show that an exact algorithm based on intelligent preprocessing, cutting planes and symmetry-breaking is capable of solving small- and medium-size instances to proven optimality, and providing strong lower bounds for larger instances.

✉ Jamie Fairbrother
j.fairbrother@lancaster.ac.uk

Adam N. Letchford
a.n.letchford@lancaster.ac.uk

Keith Briggs
keith.briggs@bt.com

[1] STOR-i Centre for Doctoral Training, Lancaster University, Lancaster, UK

[2] Department of Management Science, Lancaster University, Lancaster, UK

[3] Wireless Research Group, BT Technology, Service & Operations, Martlesham Heath, Suffolk, UK

1 Introduction

Telecommunications has proven to be a rich source of interesting optimisation problems [38]. In the case of *wireless* communications, the hardest (and most strategic) problem is *wireless network design*, which involves the simultaneous determination of cell locations and shapes, base station locations, power levels and frequency channels (e.g., [32]). On a more tactical level, one finds various *frequency assignment* problems, which are concerned solely with the assignment of available frequency bands to wireless devices (e.g., [1]).

A rather different optimisation problem arises in the context of mobile wireless communications. The technology is the 4G (LTE) standard, and the essence of the problem is as follows. There are a number of cells with known locations, and each cell must be assigned a positive integer identifier before it can operate. In the LTE standard, this is called a Physical Cell Identifier or PCI, but to keep the discussion general we will simply use the term ID. If two cells are close to each other (according to some measure of closeness), they are said to be neighbours. Two neighbouring cells must not have the same ID. We are also given two small integers k, k', both greater than two. If the IDs of two neighbouring devices are the same modulo k, it causes interference. Moreover, some additional interference occurs (but at a lesser level) if they are the same modulo kk'. Typical values are $k = 3$ and $k' = 2$, and the interferences arise because of the way reference signals (containing the ID) are encoded into a channel broadcast by the cells. The net effect is that devices wanting to connect to cells cannot do this successfully if they cannot decode the reference signals because of interference. The overall task is thus to assign IDs to cells in such a way as to minimise the total interference.

The problem turns out to be a generalisation of a well-known $\mathcal{NP}$-hard combinatorial optimisation problem called the *k-partition problem* or *k*-PP. For reasons which will become clear later, we call it the *2-level partition problem* or 2L-PP. In real-life LTE systems, the 2L-PP is typically solved approximately, via simple distributed heuristics. These heuristics perform adequately at present, but they may prove inadequate in future, as devices proliferate. To test the quality of alternative heuristics, it is necessary to have proven optimal solutions, or at least strong lower bounds, for a collection of realistic instances. For this reason, we developed an exact algorithm for the 2L-PP, based on integer programming. This algorithm turns out to be capable of solving small- and medium-sized 2L-PP instances to proven optimality, and providing strong lower bounds for larger instances.

The structure of the paper is as follows. In Sect. 2, the literature on the *k*-PP is reviewed. In Sect. 3, we formulate our problem as an integer program (IP) and derive some valid linear inequalities (i.e. cutting planes). In Sect. 4, we describe our exact algorithm in detail. In Sect. 5, we describe some computational experiments and analyse the results. Finally, some concluding remarks are made in Sect. 6.

2 Literature review

Since the 2L-PP is a generalisation of the *k*-PP, we now review the literature on the *k*-PP. We define the *k*-PP in Sect. 2.1. The main IP formulations are presented in

Sect. 2.2. The remaining two subsections cover cutting planes and algorithms for generating them, respectively.

We remark that some other multilevel graph partitioning problems have been studied in the data mining literature; see, e.g., [8,13,39,40]. In those problems, however, neither the number of clusters nor the number of levels is fixed. For this reason, we do not consider them further.

2.1 The k-partition problem

The k-PP was first defined in [7]. We are given a (simple, loopless) undirected graph G, with vertex set $V = \{1, \ldots, n\}$ and edge set E, a rational weight w_e for each edge $e \in E$, and an integer k with $2 \leqslant k \leqslant n$. The task is to partition V into k or fewer subsets, such that the sum of the weights of the edges that have both end-vertices in the same cluster is minimised. In the context of this work, it is more intuitive to interpret this as a colouring problem: each node must be assigned one of k colours, two adjacent nodes (i.e. two nodes linked by an edge) having the same colour constitutes a *conflict*, and the aim is to minimize the weighted sum of conflicts.

The k-PP has applications in scheduling, statistical clustering, numerical linear algebra, telecommunications, VLSI layout and statistical physics (see, e.g., [7,18,22, 37]). It is strongly $\mathcal{NP}$-hard for any fixed $k \geqslant 3$, since it includes as a special case the problem of testing whether a graph is k-colourable. It is also strongly $\mathcal{NP}$-hard when $k = 2$, since it is then equivalent the well-known *max-cut* problem, and when $k = n$, since it is then equivalent to the *clique partitioning* problem [24,25].

2.2 Formulations of the k-PP

Chopra and Rao [9] present two different IP formulations for the k-PP. In the first formulation, there are two sets of binary variables. For each $v \in V$ and for $c = 1, \ldots, k$, let x_{vc} be a binary variable, taking the value 1 if and only if vertex v has colour c. For each edge $e \in E$, let y_e be an additional binary variable, taking the value 1 if and only if both end-nodes of e have the same colour. Then we have the following optimization problem:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} w_e y_e \\
\text{s.t.} \quad & \sum_{c=1}^{k} x_{vc} = 1 \quad (v \in V) & (1) \\
& y_{uv} \geqslant x_{uc} + x_{vc} - 1 \ (\{u, v\} \in E, \ c = 1, \ldots, k) & (2) \\
& x_{uc} \geqslant x_{vc} + y_{uv} - 1 \ (\{u, v\} \in E, \ c = 1, \ldots, k) & (3) \\
& x_{vc} \geqslant x_{uc} + y_{uv} - 1 \ (\{u, v\} \in E, \ c = 1, \ldots, k) & (4) \\
& x_{vc} \in \{0, 1\} \quad (v \in V, \ c = 1, \ldots, k) \\
& y_{uv} \in \{0, 1\} \quad (\{u, v\} \in E).
\end{aligned}
$$

The Eq. (1) force each node to be given exactly one colour, and the constraints (2)–(4) ensure that the y variables take the value 1 when they are supposed to. Note that constraints (3) and (4) can be dropped in the case where $w_e \geqslant 0$ for all $e \in E$.

This IP has $\mathcal{O}(m + nk)$ variables and constraints, where $m = |E|$. It therefore seems suitable when k is small and G is sparse. Unfortunately, it has a very weak linear programming (LP) relaxation. Indeed, if we set all x variables to $1/k$ and all y variables to 0, we obtain the trivial lower bound of 0. Moreover, it suffers from *symmetry*, in the sense that given any feasible solution, there exist $k!$ solutions of the same cost. (See Margot [33] for a tutorial and survey on symmetry issues in integer programming.)

The second IP formulation is obtained by dropping the x variables, but having a y variable for *every* pair of nodes, setting w_{uv} to zero if $\{u, v\} \notin E$. Then:

$$\min \quad \sum_{e \in E} w_e y_e$$
$$\text{s.t.} \quad \sum_{u,v \in C} y_{uv} \geqslant 1 \quad (C \subset V : |C| = k + 1) \tag{5}$$
$$y_{uv} \geqslant y_{uw} + y_{vw} - 1 \ (\{u, v, w\} \subset V) \tag{6}$$
$$y_{uv} \in \{0, 1\} \quad (\{u, v\} \subset V).$$

The constraints (5), called *clique* inequalities, ensure that, in any set of $k + 1$ nodes, at least two receive the same colour. The constraints (6) enforce *transitivity*; that is, if nodes u and w have the same colour, and nodes v and w have the same colour, then nodes u and v must also have the same colour.

A drawback of the second IP formulation is that it has $\mathcal{O}(n^2)$ variables and $\mathcal{O}(n^{k+1})$ constraints, and it cannot exploit any special structure that G may have (such as sparsity).

A third IP formulation, based on so-called *representatives*, is studied in [2]. There also exist several *semidefinite programming* relaxations of the k-PP (see, e.g., [3,18, 20–22,37,42]). For the sake of brevity we do not go into details.

2.3 Cutting planes

Chopra & Rao [9] present several families of valid linear inequalities (i.e. cutting planes), which can be used to strengthen the LP relaxation of the above formulations. For our purposes, the most important turned out to be the *generalised clique* inequalities, which provide a lower bound on the number of conflicts for any clique with more than k nodes. In the case of the first IP formulation, they take the form

$$\sum_{u,v \in C} y_{uv} \geqslant \binom{t + 1}{2} r + \binom{t}{2} (k - r), \tag{7}$$

where $C \subseteq V$ is a clique (set of pairwise adjacent nodes) in G with $|C| > k$, and t and r denote $\lfloor |C|/k \rfloor$ and $r = |C| \bmod k$, respectively. In the case of the second IP formulation, they must be defined for any $C \subseteq V$ with $|C| > k$ (since every set of nodes forms a clique in a complete graph). In either case, they define facets of the associated polytope when $k \geqslant 3$ and $r \neq 0$. Note that they reduce to the clique inequalities (5) when $|C| = k + 1$.

Further inequalities for the first IP formulation can be found in [9,20]. Further inequalities for the second formulation can be found in, e.g., [9,10,15,16,25,36].

2.4 Separation algorithms

For a given family of valid inequalities, a *separation algorithm* is an algorithm which takes an LP solution and searches for violated inequalities in that family [23].

By brute-force enumeration, one can solve the separation problem for the inequalities (2)–(4) in $\mathcal{O}(km)$ time, for the transitivity inequalities (6) in $\mathcal{O}(n^3)$ time, and for the clique inequalities (5) in $\mathcal{O}(n^{k+1})$ time. It is stated in [9] that separation of the generalised clique inequalities (7) is $\mathcal{NP}$-hard. An explicit proof, using a reduction from the max-clique problem, is given in [17]. Heuristics for clique and generalised clique separation are presented in [17,29].

Separation results for other inequalities for the first IP formulation can be found in [9,20]. Separation results for the second formulation can be found in, e.g.,[4,6,16, 24,30,35,36]. For some computational results with various separation algorithms, see [14].

3 Formulation and valid inequalities

In this section, we give an IP formulation of the 2L-PP (Sect. 3.1) and derive some valid inequalities (3.2). We also show how to modify the formulation to address issues of *symmetry* (3.3).

3.1 Integer programming formulation

In order to demonstrate that the 2L-PP is a generalisation of the k-PP, we present the problem in its most general form, before describing how this is used to model our telecommunications problem. An instance of the 2L-PP is given by an undirected graph $G = (V, E)$, integers $k, k' \geqslant 2$, and two sets of edge weights $w_e, w'_e \in \mathbb{R}$ for $e \in E$. The 2L-PP is a type of node colouring problem with colours $\{0, \ldots, kk' - 1\}$ and where two types of conflicts may occur: a *modulo k conflict* occurs if two adjacent nodes are assigned the same colour up to modulo k, and a *modulo kk' conflict* occurs if two adjacent nodes use exactly the same colour. The aim of the 2L-PP is to find a colouring which minimizes the sum of modulo k conflicts weighted by $(w_e)_{e \in E}$, plus the sum of modulo kk' conflicts weighted by $(w'_e)_{e \in E}$.

To formulate the 2L-PP as an IP, we modify the first formulation mentioned in Sect. 2.2. We have three sets of binary variables. For each $v \in V$ and for $c = 0, \ldots, kk' - 1$, let x_{vc} be a binary variable, taking the value 1 if and only if vertex v has colour c. For each edge $e \in E$, define two binary variables y_e and z_e, taking the value 1 if and only if both end-nodes of e have the same colour modulo k, or the same colour, respectively. Then we have:

$$\min \quad \sum_{\{u,v\}\in E} \left(w_e y_{uv} + w'_e z_{uv} \right) \tag{8}$$

$$\text{s.t.} \quad \sum_{c=0}^{kk'-1} x_{vc} = 1 \qquad (v \in V) \tag{9}$$

$$y_{uv} \geqslant \sum_{r=0}^{k'-1} x_{u,c+rk} + \sum_{r=0}^{k'-1} x_{v,c+rk} - 1 \; (\{u,v\} \in E, \; c = 0, \ldots, k-1) \tag{10}$$

$$\sum_{r=0}^{k'-1} x_{u,c+rk} \geqslant y_{uv} + \sum_{r=0}^{k'-1} x_{v,c+rk} - 1 \; (\{u,v\} \in E, \; c = 0, \ldots, k-1) \tag{11}$$

$$\sum_{r=0}^{k'-1} x_{v,c+rk} \geqslant y_{uv} + \sum_{r=0}^{k'-1} x_{u,c+rk} - 1 \; (\{u,v\} \in E, \; c = 0, \ldots, k-1) \tag{12}$$

$$z_{uv} \geqslant x_{uc} + x_{vc} - 1 \qquad (\{u,v\} \in E, \; c = 0, \ldots, kk'-1) \tag{13}$$

$$x_{uc} \geqslant z_{uv} + x_{vc} - 1 \qquad (\{u,v\} \in E, \; c = 0, \ldots, kk'-1) \tag{14}$$

$$x_{vc} \geqslant z_{uv} + x_{uc} - 1 \qquad (\{u,v\} \in E, \; c = 0, \ldots, kk'-1) \tag{15}$$

$$x_{vc} \in \{0,1\} \qquad (v \in V, \; c = 0, \ldots, kk'-1) \tag{16}$$

$$y_{uv}, z_{uv} \in \{0,1\} \qquad (\{u,v\} \in E). \tag{17}$$

The objective function (8) is just a weighted sum of the two kinds of colour conflicts. The constraints (9) state that each node must have a unique colour. The constraints (10)–(12) ensure $(y_e)_{e\in E}$ indicate modulo k colour conflicts while (13)–(15) ensure $(z_e)_{e\in E}$ indicate modulo kk' conflicts. The remaining constraints are just binary conditions. In the above formulation, when $k' = 1$ the constraints (10)–(15) force the variables y_e and z_e to be equal for all $e \in E$, and so the problem becomes a k-PP.

Our telecommunications application is modelled by the 2L-PP as follows: each node in V corresponds to a device, and a pair of nodes is connected by an edge if and only if the corresponding devices are neighbours; assigning the colour c to a node corresponds to giving the corresponding device an ID that is congruent to c modulo kk'; the weights of each type of conflict constant and are given by positive numbers w and w' which specify the relative importance given to interference modulo k and modulo kk', respectively; finally the aim of the problem is to minimize the weighted sum of the two kinds of interference. As in the case of the k-PP, the positive edge weights render some of the constraints of the above formulation redundant, in particular (11), (12), (14) and (15). Note that the above IP has $kk'n + 2m$ variables and $n + 3k(k'+1)m$ linear constraints and so for sparse graphs and small values k and k', which characterise our application, this formulation is manageable.

An intuitive approach to solving the 2L-PP would be to first solve the k-PP, and then solve the k'-PP for each of the k subgraphs induced by nodes of the same colour, using appropriate weights at each stage. However, this method does not necessarily yield an optimal solution with respect to the above formulation even in the case where we have constant positive edge weights as in our application. This is because there

may exist substantial flexibility in the topologies of the k induced subgraphs which minimize the number of modulo k conflicts, and these different topologies may have different optimal numbers of k' conflicts. Therefore, optimizing modulo k conflicts first will not necessarily yield subgraphs which produce an optimal weighted sum of modulo k and kk' conflicts. In the case where edge weights vary independently for each type of conflict, this heuristic may perform even worse as a subgraph induced by nodes of the same colour modulo k which has small modulo k weights, may have large modulo kk' weights.

3.2 Valid inequalities

Unfortunately, our IP formulation of the 2L-PP shares the same drawbacks as the first formulation of the k-PP mentioned in Sect. 2.2: it has a very weak LP relaxation (giving a trivial lower bound of zero), and it suffers from a high degree of symmetry.

To strengthen the LP relaxation, we add valid linear inequalities from three families. For a clique $C \subset V$, let $y(C)$ and $z(C)$ denote $\sum_{\{u,v\} \subset C} y_{uv}$ and $\sum_{\{u,v\} \subset C} z_{uv}$, respectively. The first two families of inequalities are straightforward adaptations of the generalised clique inequalities (7) for the k-PP. Specifically, given a clique C, the inequalities (18) and (19) give lower bounds on the number of modulo k and modulo kk' conflicts which any colouring must induce.

Proposition 1 *The following inequalities are satisfied by all feasible solutions of the 2L-PP:*

- *"y-clique" inequalities, which take the form:*

$$y(C) \geqslant \binom{t+1}{2} r + \binom{t}{2} (k - r), \tag{18}$$

 where $C \subseteq V$ is a clique with $|C| > k$, $t = \lfloor |C|/k \rfloor$ and $r = |C| \bmod k$;
- *"z-clique" inequalities, which take the form:*

$$z(C) \geqslant \binom{T+1}{2} R + \binom{T}{2} (kk' - R), \tag{19}$$

 where $C \subseteq V$ is a clique with $|C| > kk'$, $T = \lfloor \frac{|C|}{kk'} \rfloor$ and $R = |C| \bmod kk'$.

Proof This follows from the result of Chopra & Rao [9] mentioned in Sect. 2.3, together with the fact that, in a feasible IP solution, the y and z vectors are the incidence vectors of a k-partition and a kk'-partition, respectively. □

The third family of inequalities, which is completely new, is described in the following theorem. These inequalities follow from the fact that a kk' colouring induces a partition of a clique into k subcliques of the same colours modulo k, and each of these subcliques must have at least a given number of modulo k' conflicts.

Theorem 1 *For all cliques $C \subseteq V$ with $|C| > k'$, the following "(y, z)-clique" inequalities are valid:*

$$k' z(C) \geqslant y(C) - t' \binom{k'}{2} - \binom{r'}{2}, \tag{20}$$

where $t' = \lfloor |C|/k' \rfloor$ and $r' = |C| \bmod k'$.

Proof See the "Appendix".   □

The following two lemmas and theorem give necessary conditions for the inequalities presented so far to be facets (i.e., not implied by other inequalities).

Lemma 1 *A necessary condition for the y-clique inequality* (18) *to be a facet is that $r \neq 0$.*

Proof This was already shown by Chopra and Rao [9].   □

Lemma 2 *A necessary condition for the (y, z)-clique inequality* (20) *to be facet is that $r' \neq 0$.*

Proof Suppose that $r' = 0$. The (y, z)-clique inequality for C can be written as:

$$k' z(C) \geqslant y(C) - |C|(k' - 1)/2. \tag{21}$$

Now let v be an arbitrary node in C. The (y, z)-clique inequality for the set $C \setminus \{v\}$ can be written as:

$$k' z(C \setminus \{v\}) \geqslant y(C \setminus \{v\}) - (|C| - 2)(k' - 1)/2.$$

Summing this up over all $v \in C$ yields

$$k'(|C| - 2) z(C) \geqslant (|C| - 2) y(C) - |C|(|C| - 2)(k' - 1)/2.$$

Dividing this by $|C| - 2$ yields the inequality (21).   □

Theorem 2 *A necessary condition for the z-clique inequality* (19) *to be a facet is that $1 < R < kk' - 1$.*

Proof See the "Appendix".   □

Our experiments with the polyhedron transformation software package PORTA [11] lead us to make the following conjecture:

Conjecture 1 *The following results hold for the convex hull of 2L-PP solutions:*

- *y-clique inequalities* (7) *define facets if and only if $r \neq 0$.*
- *z-clique inequalities* (19) *define facets if and only if $1 < R < kk' - 1$.*
- *(y, z)-clique inequalities* (20) *define facets if and only if $r' \neq 0$.*

In any case, we have found that all three families of inequalities work very well in practice as cutting planes. Moreover, in our preliminary experiments, we found that the y-clique inequalities were the most effective at improving the lower bound, with the yz-clique inequalities being the second most effective.

Remark The trivial inequality $y_e \geqslant z_e$ is also valid for all $e \in E$. In our preliminary experiments, however, these inequalities proved to be of no value as cutting planes.

3.3 Symmetry

Another issue to address is *symmetry*. Note that any permutation σ on the set of colours $\{0, \ldots, kk' - 1\}$ such that $\sigma(c) \bmod k = c \bmod k$ will preserve all k and kk' conflicts. Since there are $k'!$ such permutations, for any colouring (which makes use of all available colours) there are at least $k'!$ other colourings which yield the same cost.

One easy way to address this problem, at least partially, is given in the following theorem:

Theorem 3 *For any colour $c \in \{0, \ldots, kk' - 1\}$, let $\phi(c)$ denote $\lfloor c/k \rfloor + (c \bmod k)$. Then, one can fix to zero all variables x_{vc} for which $\phi(c) \geqslant v$, while preserving at least one optimal 2L-PP solution.*

Proof See the "Appendix". □

Example Suppose that $n \geqslant 4$ and $k = k' = 3$. Then $\phi(0), \ldots, \phi(8)$ are 0, 1, 2, 1, 2, 3, 2, 3 and 4, respectively. So we can fix the following variables to zero: $x_{11}, \ldots, x_{18}$; x_{22}; $x_{24}, \ldots, x_{28}$; x_{35}, x_{37}, x_{38} and x_{48}. □

4 Exact algorithm

We now describe an exact solution algorithm for the 2L-PP. The algorithm consists of two main stages: *preprocessing* and *cut-and-branch*. Preprocessing is described in Sect. 4.1, while the cut-and-branch algorithm is described in Sect. 4.2. Throughout this section, for a given set of nodes $V' \subseteq V$, we let $G[V']$ denote the subgraph of G induced by the nodes in V'.

4.1 Preprocessing

In the first stage, an attempt is made to simplify the input graph G and, if possible, decompose it into smaller and simpler subgraphs. This is via two operations, which we call *k-core reduction* and *block decomposition*. Although we focus on the 2L-PP the following results also apply to the k-PP.

A k-core of a graph G is a maximal connected subgraph whose nodes all have degree of at least k. The concept was first introduced in [41], as a tool to measure cohesion in social networks. An example is given in Fig. 1, but it should be borne in mind that, in general, a graph may have several (node-disjoint) k-cores. The k-cores
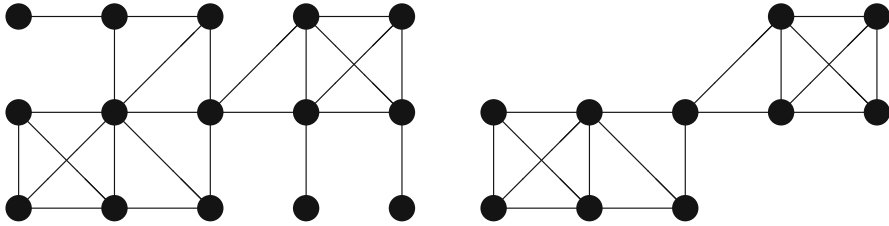
Fig. 1 A graph (left) and its (unique) 3-core (right)

of a graph can be found easily, in $\mathcal{O}(n^2)$ time, via a minor adaptation of an algorithm given in [43]. Details are given in Algorithm 1.

Algorithm 1: Algorithm for finding k-cores of a graph

input : graph $G = (V, E)$
output: k-cores of G
$V' := V$;
do
    Let V^- equal $\{v \in V' : \deg_{G[V']}(v) < k\}$;
    if $V^- \neq \emptyset$ **then**
       | $V' := V' \setminus V^-$;
    end
while $V^- \neq \emptyset$;
Output the connected components of $G[V']$

The reason that k-cores are of interest is given in the following proposition:

Proposition 2 *For any graph G, and non-negative edge weights w_e and w'_e for $e \in E$, the cost of the optimal 2L-PP solution is equal to the sum of the costs of the optimal solutions of the 2L-PP instances given by its k-cores.*

Proof (sketch) Let $v \in V$ be any node whose degree in G is less than k. Suppose we solve the 2L-PP on the induced subgraph $G[V \setminus \{v\}]$. Then we can extend the 2L-PP solution to the original graph G, without increasing its cost, by giving node v a colour that is not congruent modulo k to the colour of any of its neighbours. The result follows by induction.   □

We refer to the process of the replacement of G with its k-core(s) as *k-core reduction*. We will say that a graph is *k-core reducible* if it is not equal to the union of its k-cores.

Now, a vertex of a graph is said to be an *articulation point* if its removal causes the graph to become disconnected. A connected graph with no articulation points is said to be *biconnected*. The *biconnected components* of a graph, also called *blocks*, are maximal induced biconnected subgraphs. For example, the graph on the right of Fig. 1 has two blocks, which are displayed on the left of Fig. 2. The blocks of a graph $G = (V, E)$ can be computed in $\mathcal{O}(|V| + |E|)$ time [28].

It has been noted that many optimisation problems on graphs can be simplified by working on the blocks of the graph instead of the original graph; see, e.g., [27]. The following proposition shows that this is also the case for the 2L-PP.
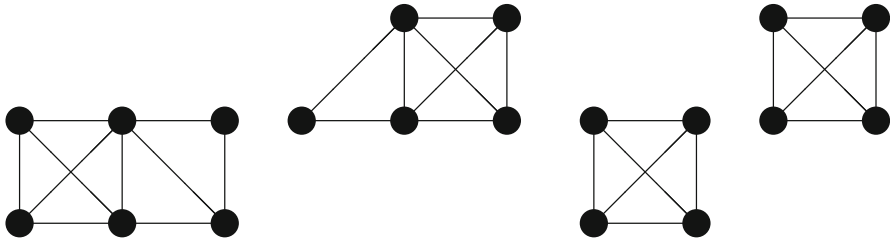
Fig. 2 Blocks of the 3-core (left) and 3-cores of the blocks (right)

Proposition 3 *For any graph G, the cost of the optimal 2L-PP solution is equal to the sum of the costs of the optimal solutions of the 2L-PP instances given by its blocks.*

Proof (sketch) If G is disconnected, then the 2L-PP trivially decomposes into one 2L-PP instance for each connected component. So assume that G is connected but not biconnected. Let $v \in V$ be an articulation point, and let $V_1, \ldots, V_t$ be the vertex sets of the connected components of $G[V \setminus \{v\}]$. Now consider the subgraphs $G[V_i \cup \{v\}]$ for $i = 1, \ldots, t$. Let $S(i)$ be the optimal solution to the 2L-PP instance on $G[V_i \cup \{v\}]$, represented as a proper kk'-colouring of $V_i \cup \{v\}$, and let $c(i)$ be its cost. Since each edge of E appears in exactly one of the given subgraphs, the quantity $\sum_{i=1}^{t} c(i)$ is a lower bound on the cost of the optimal 2L-PP solution on G. Moreover, by symmetry, we can assume that node v receives colour 1 in $S(1), \ldots, S(t)$. Now, for a given $u \in V \setminus \{v\}$, let $i(u) \in \{1, \ldots, t\}$ be the unique integer such that $u \in V_{i(u)}$. We can now construct a feasible solution to the 2L-PP on G by giving node v colour 1, and giving each other node $u \in V \setminus \{v\}$ the colour that it has in $S(i(u))$. The resulting 2L-PP solution has cost equal to $\sum_{i=1}^{t} c(i)$, and is therefore optimal. $\qquad\square$

We call the replacement of a graph with its blocks *block decomposition*. Interestingly, a graph which is not k-core reducible may have blocks which are; see again Fig. 2. This leads us to apply k-core reduction and block decomposition recursively, until no more reduction or decomposition is possible.

At the end of this procedure, we have a collection of induced subgraphs of G which are biconnected and not k-core reducible, and we can solve the 2L-PP on each subgraph independently. Given the optimal solutions for each subgraph, we can reconstruct an optimal solution for the original graph by recursively constructing solutions for the predecessor graph of a reduction or decomposition.

4.2 Cut-and-branch algorithm

For each remaining subgraph, we now run our cut-and-branch algorithm. Let $G' = (V', E')$ be the given subgraph. We set up an initial trivial LP, with only one variable y_e for each edge $e \in E'$, and run a cutting-plane algorithm based on y-clique inequalities. Next, we add the z variables and run another cutting-plane algorithm based on z-clique and yz-clique inequalities. Finally, we add the x-variables and run branch-and-bound. The full procedure is detailed in Algorithm 2.

Algorithm 2: Cut-and-branch algorithm to solve 2L-PP

---

input : subgraph G', 2L-PP problem parameters k, k', continuous parameter $\epsilon > 0$, integer
parameter $t > 0$

Enumerate all cliques in G' of size greater than k;
Construct the (trivial) LP relaxation min $\sum_{e \in E'} y_e$ s.t. y non-negative;
do
    Solve LP relaxation;
    Search for violated y-clique inequalities (18);
    If any are found, add the t most violated ones to the LP;
while *Violated inequalities found*;
Delete all y-clique inequalities with slack greater than ϵ;
Change objective function to (8) and add one non-negative variable z_e for all $e \in E'$;
do
    Solve LP relaxation;
    Search for violated z-clique inequalities (19) and yz-clique inequalities (20);
    If any are found, add the t most violated ones to the LP;
while *Violated inequalities found*;
Delete all inequalities with slack greater than ϵ;
Add x variables and constraints (9)–(17), and apply symmetry-breaking;
Solve resulting 0-1 LP with branch-and-bound;

The key feature of this approach is that the LP is kept as small as possible throughout the course of the algorithm. Indeed, (a) the z and x variables are added to the problem only when they are needed, (b) only a limited number of constraints are added in each cutting-plane iteration, and (c) slack constraints are deleted after each of the two cutting-plane algorithms has terminated. The net result is that both cutting-plane algorithms run very efficiently, and so does the branch-and-bound algorithm at the end.

We now make some remarks about the separation problems for the three kinds of clique inequalities. Since all three separation problems seem likely to be $\mathcal{NP}$-hard, we initially planned to use greedy separation heuristics, in which the set C is enlarged one node at a time. We were surprised to find, however, that it was feasible to solve the separation problems exactly, by brute-force enumeration, for typical 2L-PP instances encountered in our application. The reason is that the original graph G tends to be fairly sparse in practice, and each subgraph G' generated by our preprocessor tends to be fairly small. Accordingly, after the preprocessing stage, we use the Bron–Kerbosch algorithm [5] to enumerate all maximal cliques in each subgraph G'. It is then fairly easy to solve the separation problems by enumeration, provided that one takes care not to examine the same clique twice in a given separation call. We omit details, for brevity.

We remark that, although an arbitrary graph with n nodes can have as many as $3^{\frac{n}{3}}$ maximal cliques [34], a graph in which all nodes have degree at most d can have at most $(n - d)3^{\frac{d}{3}}$ of them [19].

5 Computational experiments

We now present the results of some computational experiments. In Sect. 5.1, we describe how we constructed the graphs used in our experiments. In Sect. 5.2, we test

the preprocessing algorithm for different values of k. In Sect. 5.3 we study the effect of our symmetry-breaking constraints on the required solution time. In Sect. 5.4, we present results on our the cut-and-branch algorithms.

Throughout these experiments, the value of k varies between 2 and 5, while for simplicity, we fix $k' = 2$, since the value of k' does not affect the performance of the graph preprocessing algorithm. All experiments have been run on a high performance computer with an Intel 2.6 GHz processor and using 16 cores. Graph preprocessing and clique enumeration was done using `igraph` [12] and the linear and integer programs were solved using `Gurobi` v.7.5 [26]. All ILP instances are given a time limit of 3 h to be solved. Unless otherwise stated we use the default for all other of Gurobi's settings.

5.1 Graph construction

The strength of a signal at a receiver decays in free space at a rate inversely proportional to the square of the distance from the transmitter, but in real systems often at a faster rate due to the presence of objects blocking or scattering the waves. Therefore, beyond a certain distance, two transceivers can no longer hear each other, and therefore there cannot be a direct conflict between their IDs. In our application, however, a conflict also occurs if a pair of devices have a neighbour in common. Essentially, this is because each device needs to be able to tell its neighbours apart.

Accordingly, we initially constructed our graphs as follows. We first sample a specified number of points uniformly on the unit square. Edges are created between pairs of points if they are within a specified radius of each other. This yields a so-called *disk graph*; see, e.g., [31]. The graph is then augmented with edges between pairs of nodes which have a neighbour in common. (In other words, we take the *square* of the disk graph.) This construction is illustrated in Fig. 3.

It turned out, however, that neighbourhood graphs constructed in this way yielded extremely easy 2L-PP instances. The reason is that nodes near to the boundary of
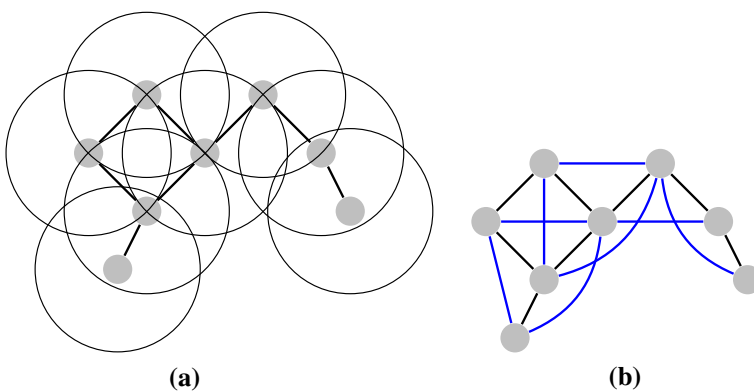

Fig. 3 Construction of neighbourhood graph. **a** Edges are created between nodes within a given radius of each other. **b** Graph is augmented with edges between nodes which have a neighbour in common
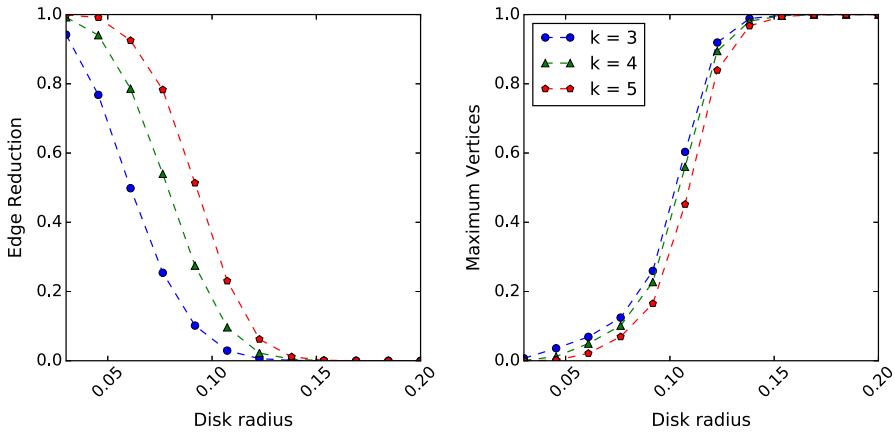
Fig. 4 Preprocessing of random neighbourhood graphs

the square tend to have small degree, which causes them to be removed during the preprocessing stage. This in turn causes their neighbours to have small degree, and so on. In order to create more challenging instances, and to avoid this "boundary effect", we decided to use a torus topology to calculate distances between points in the unit square before constructing the graphs.

5.2 Preprocessing

In order to understand the potential benefits of the preprocessing stage, we have calculated the effect of preprocessing on our random neighbourhood graphs for different values of k and disk radius, while fixing $n = 100$. In particular, for radius $0.01, \ldots, 0.2$ and $k = 3, 4, 5$, we calculate the mean proportion of edges eliminated and the mean proportion of vertices in the largest remaining component. The results are shown in Fig. 4. The means were estimated by simulating 1000 random graphs for each pair of parameters.

The results show that, for all values of k, the proportion of edges eliminated decays to zero rather quickly as the disk radius is increased. The proportion of nodes in the largest component also tends to one as the radius is increased, but at a slower rate than the convergence for edge reduction.

In order to gain further insight, we also explored how the average degree in our random neighbourhood graphs depends on the number of nodes and the disk radius. The results are shown in Fig. 5. A comparison of this figure and the preceding one indicates that, as one might expect, preprocessing works best when the average degree is not much larger than k.

5.3 Symmetry constraints

In this numerical experiment we gauge the utility of our symmetry-breaking constraints by looking at the branch-and-bound time required to solve the 2L-PP. In particular,
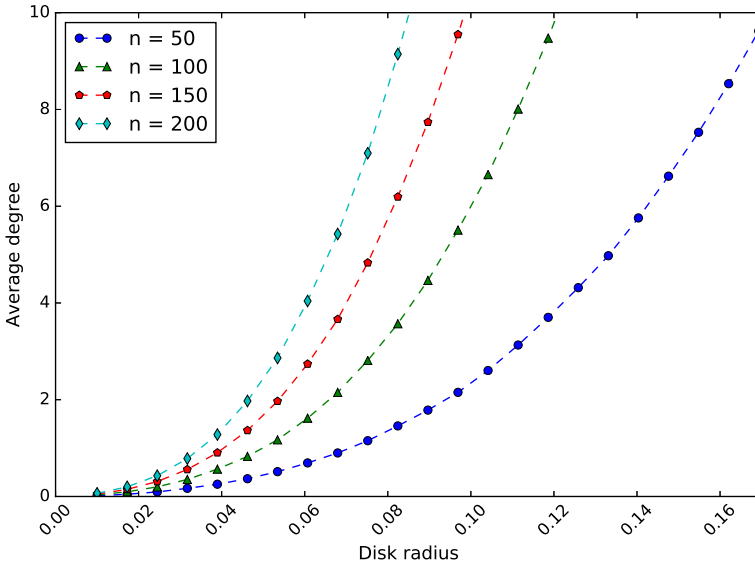
Fig. 5 Mean average degree of random neighbourhood graphs

Table 1 Comparison of average branch-and-bound times (seconds) using Gurobi's symmetry detection capabilities (SymDet), our symmetry-breaking constraints (SymCons) and neither (NoSym)

Radius	k = 2			k = 3		
	SymDet	SymCons	NoSym	SymDet	SymCons	NoSym
0.08	0.07	0.10	0.11	0.05	0.06	0.07
0.09	0.14	0.21	0.20	0.11	0.14	0.14
0.10	0.46	0.78	0.78	0.38	0.56	2.00
0.11	2.07	5.07	7.16	2.07	4.46	143.55
0.12	25.96	46.35	105.63	50.97	–	–

we compare the times required for solution using our symmetry-breaking constraints, using Gurobi's symmetry detection capabilities, and using neither symmetry detection nor symmetry-breaking constraints.

For $n = 100$, and for disk radii ranging from 0.08 to 0.12, we construct 30 random neighbourhood graphs. The same sets of points were used to construct the graphs for each disk radius. We also considered two values of k, namely 2 and 3. The weights w, w' were both set to 1 for simplicity. For each of the resulting 300 2L-PP instances, we run branch-and-bound with preprocessing but no cutting planes. When solving the problems using our symmetry-breaking constraints, we disable Gurobi's symmetry detection capabilities.

The results are shown in Table 1. A dash means that at least one instance failed to solve for that radius and value of k. In this experiment one instance out of the thirty failed to solve within the given time limit for $r = 0.12$ and $k = 3$ when we used our

symmetry-breaking constraints, and four failed to solve for the same radius when nei-
ther symmetry-breaking constraint nor symmetry detection were used. These results
demonstrate that using our symmetry-breaking constraints, in general, significantly
reduces the solution time required to solve the problem compared to using the Gurobi
solver with symmetry detection capabilities switched off, and this benefit increases
as the graphs become dense. However, Gurobi's symmetry detection capabilities gen-
erally outperform just using our symmetry-breaking constraints. This indicates that
these constraints will generally only be useful for MIP solvers which do not have
sophisticated symmetry-detection algorithms. In order to test our methodology for
more dense graphs, we will therefore use Gurobi's symmetry detection capabilities
rather than our symmetry-breaking constraints.

5.4 Cutting planes

Next, we present two numerical experiments on the effectiveness of using our cutting
planes. In first experiment, we compare the branch-and-bound time required to solve
the 2L-PP problems using and not using our cutting plane algorithm beforehand. We
solve the exact same 300 problems as were used in the previous experiment, and
preprocess the graphs before solving the problem. Note that we do not include the
time of the cutting plane algorithm in this comparison. As the next experiment shows,
the cutting plane time is negligible compared to the branch-and-bound time.

The results for this first experiment are shown in Table 2. These demonstrate that
using our cutting planes reduces the solution time required dramatically, and by an
order of magnitude for harder problems.

In the final experiment, we study the lower bounds yielded by each phase of the
cutting plane algorithm, and the times these take to run. In this experiment, we use the
previously constructed problems, and in addition construct more difficult problems,
which may not have been solvable in the previous experiments. For the disk radii
0.13 and 0.14 we construct another 30 problems. In addition, we also solve the 2L-PP
for $k = 4$ for all constructed graphs, as this problem will be non-trivial for the new
denser graphs. For each of the resulting 630 2L-PP instances, we run the preprocessing
algorithm, followed by the cut-and-branch algorithm.

In order to measure the effects of the yz-clique and the z-clique inequalities, we
have divided the second cutting-plane phase into two sub-phases, whereby yz-clique

Table 2 Comparison of average branch-and-bound times (in seconds) required to solve problem using our cutting plane algorithm beforehand (Cuts) and not using cuts (NoCuts)

Radius	k = 2		k = 3	
	Cuts	No cuts	Cuts	No cuts
0.08	0.02	0.07	0.01	0.05
0.09	0.04	0.15	0.02	0.11
0.10	0.09	0.46	0.03	0.38
0.11	0.19	2.07	0.16	2.09
0.12	0.93	25.50	0.37	51.69

Table 3 Average relative optimality gap at the end of each cutting-plane phase

Radius	k = 2			k = 3			k = 4		
	y-cut gap	yz-cut gap	z-cut gap	y-cut gap	yz-cut gap	z-cut gap	y-cut gap	yz-cut gap	z-cut gap
0.08	0.03	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.09	0.05	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00
0.10	0.07	0.02	0.01	0.01	0.01	0.01	0.00	0.00	0.00
0.11	0.09	0.02	0.01	0.05	0.04	0.04	0.04	0.04	0.04
0.12	0.11	0.02	0.01	0.05	0.03	0.03	0.02	0.02	0.02
0.13	0.15	0.04	0.03	0.07	0.04	0.04	0.03	0.03	0.03
0.14	0.16	0.04	0.03	–	–	–	0.04	0.04	0.04

Table 4 Average number of cuts added to linear relaxation during each cutting-plane phase

Radius	k = 2			k = 3			k = 4		
	# y-cuts	# yz-cuts	# z-cuts	# y-cuts	# yz-cuts	# z-cuts	# y-cuts	# yz-cuts	# z-cuts
0.08	45.4	1.2	0.4	9.3	0.0	0.0	1.2	0.0	0.0
0.09	73.9	3.0	0.8	18.4	0.0	0.0	2.9	0.0	0.0
0.10	108.1	7.8	2.9	32.7	0.1	0.0	7.5	0.0	0.0
0.11	159.5	16.8	7.1	57.9	0.4	0.1	15.1	0.0	0.0
0.12	203.3	29.6	13.7	84.5	1.3	0.2	25.3	0.0	0.0
0.13	274.2	53.7	21.2	119.8	2.8	0.7	39.5	0.0	0.0
0.14	373.8	86.5	30.6	178.2	6.4	2.5	61.4	0.1	0.0

inequalities are added in the first subphase and z-clique inequalities are added in the second subphase.

In Table 3, we present the average gap between the lower bound and optimum, expressed as a proportion of the optimum, after each of the three kinds of cuts have been added. Note that for $r = 0.14$ and $k = 3$ one instance of the thirty failed to solve. From these results we see that, for each value of k, the gap increases as we enlarge the disk radius. Nevertheless, the gap always remains below 5% for the radii considered. Note that the z-clique inequalities help only when $k = 2$. This may be because z-clique inequalities are defined only for cliques containing more than kk' nodes, and not many such cliques are present when $k > 2$.

In Table 4 we present the average number of cuts added during each cutting-plane phase. We see that the number of cuts increases with the disk radius. This is to be expected, since, when the radius is large, there are more large cliques present. On the other hand, the number of cuts decreases as the value of k increases. The explanation for this is that y-clique inequalities are defined only for cliques containing more than k nodes, and the number of such cliques decreases as k increases.

Finally, Table 5 presents the running times of the cutting plane and branch-and-bound phases of the algorithm. As expected, the running time increases as the disk

Table 5 Average cutting plane (CP) and branch-and-bound (BB) times

Radius	k = 2		k = 3		k = 4	
	CP time (s)	BB time (s)	CP time (s)	BB time (s)	CP time (s)	BB time (s)
0.08	0.01	0.02	0.00	0.01	0.00	0.00
0.09	0.01	0.04	0.00	0.02	0.00	0.00
0.10	0.02	0.09	0.00	0.03	0.00	0.01
0.11	0.02	0.19	0.01	0.16	0.00	0.04
0.12	0.04	0.93	0.01	0.37	0.00	0.10
0.13	0.07	5.91	0.02	13.64	0.01	0.36
0.14	0.15	74.28	0.04	–	0.01	4.47

radius increases and the graph becomes more dense. On the other hand, perhaps surprisingly, it decreases as k increases. This is partly because the preprocessing stage removes more nodes and edges when k is larger but also because fewer edge conflicts occur when we use more colours. We also see that, for all values of k and disk radii, the running time of the cutting-plane stage is negligible compared with the running time of the branch-and-bound stage. This is so, despite the fact that we are using enumeration to solve the separation problems exactly.

6 Conclusions

In this paper we have defined and tackled the 2-level graph partitioning problem which, as far as we are aware, has not previously been addressed in the optimization or data-mining literature. Although this model was motivated by a problem in telecommunications it may have other applications, such as in hierarchical clustering.

The instances encountered in our application were characterised by small values of k and k', and large, sparse graphs. For instances of this kind, we proposed a solution approach based on aggressive preprocessing of the original graph, followed by a novel multi-layered cut-and-branch scheme, which is designed to keep the LP as small as possible at each stage. Along the way, we also derived new valid inequalities and symmetry-breaking constraints.

One possible topic for future research is the derivation of additional families of valid inequalities, along with accompanying separation algorithms (either exact or heuristic). Another interesting topic is the "dynamic" version of our problem, in which devices are switched on or off from time to time.

Appendix

Proof of Theorem 1 For a clique $C \subset V$ let $\mathcal{C} : C \to \{0, \ldots, kk'-1\}$ be a kk'-coloring of C. For $c = 0, \ldots, k-1$, let S_c denote $\{v \in C : \mathcal{C}(v) \in \{c, c+k, \ldots, c+(k'-1)k\}\}$ and for $c = 0, \ldots, kk' - 1$ let W_c denote $\{v \in C : \mathcal{C}(v) = c\}$. Note that, for $c = 0, \ldots, k-1$,

$$\bigcup_{t=0}^{k'-1} W_{c+tk} = S_c. \tag{22}$$

Fix $c \in \{0, \ldots, k-1\}$. By definition we have

$$y(S_c) = \binom{|S_c|}{2},$$

and

$$z(S_c) = \sum_{t=0}^{k'-1} \binom{|W_{c+tk}|}{2}.$$

Suppose $|S_c| = p_c k' + r_c$ where $0 \leqslant r_c < k'$. Then the second summation is minimized when

$$|W_{c+tk}| = p_c + 1 \qquad \text{for } t = 0, \ldots, r-1$$
$$= p_c \qquad \text{for } t = r, \ldots, k'-1.$$

Hence,

$$z(S_c) \geqslant r_c \binom{p_c + 1}{2} + (k' - r_c)\binom{p_c}{2}$$

$$= \frac{1}{2}\left(k' p_c^2 + 2p_c r_c - k' p_c\right)$$

$$= \frac{1}{2k'}\left(k'^2 p_c^2 + 2k' p_c r_c - k'^2 p_c\right)$$

$$= \frac{1}{2k'}\left((k' p_c + r)(k' p_c + r - 1) + k' p_c + r_c - r_c^2 + k'^2 p_c\right)$$

$$= \frac{1}{k'}\binom{|S_c|}{2} + \frac{1}{2k'}\left(|S_c| - (k'^2 p_c + r_c^2)\right)$$

$$= \frac{1}{k'} y(S_c) + \frac{1}{2k'}\left(|S_c| - \left(k'(|S_c| - r_c) + r_c^2\right)\right)$$

$$= \frac{1}{k'} y(S_c) - \frac{1}{2k'}\left((k' - 1)|S_c| - r_c(k' - r_c)\right).$$

Now,

$$
\begin{aligned}
z(C) &= \sum_{c=0}^{k-1} z(S_c) \\
&\geqslant \sum_{c=0}^{k-1} \left(\frac{1}{k'} y(S_c) - \frac{1}{2k'} \left((k'-1)|S_c| - r_c(k'-r_c) \right) \right) \\
&\geqslant \frac{1}{k'} y(C) - \frac{1}{2k'} \left((k'-1)|C| - \sum_{c=0}^{k-1} r_c(k'-r_c) \right).
\end{aligned}
$$

The last expression is minimized when $\sum_{c=0}^{k-1} r_c(k'-r_c)$ is minimized. Noting that $\sum_{c=0}^{k-1} r_c \bmod k' \geqslant |C| \bmod k' = R$, we see that this expression is minimized when $r_0 = R$ and $r_c = 0$ for $c = 1, \ldots, k-1$. Hence,

$$
z(C) \geqslant \frac{1}{k'} y(C) - \frac{1}{2k'} \left((k'-1)|C| - R(k'-R) \right),
$$

which is equivalent to the (y, z)-clique inequality (20). $\square$

Proof of Theorem 2 Let $R = |C| \bmod kk'$. For the case $R = 0$, the z-clique inequality is implied by other z-clique inequalities (see [9]). For the cases $R = 1$ and $R = kk'-1$, we show that the associated z-clique inequality is implied by y-clique and (y, z)-clique inequalities.

First, suppose that $R = 1$, i.e., that $|C| = Tkk' + 1$ for some positive integer T. In this case, the z-clique inequality takes the form:

$$
z(C) \geqslant \binom{T+1}{2} + \binom{T}{2}(kk'-1) = k'(k/2)(T^2 - T) + T. \tag{23}
$$

The y-clique inequality on C takes the form:

$$
y(C) \geqslant \binom{k'T+1}{2} + \binom{k'T}{2}(k-1) = k'(k/2)(k'T^2 - T) + k'T, \tag{24}
$$

and the (y, z)-clique inequality on C takes the form:

$$
k'z(C) - y(C) \geqslant -Tk\binom{k'}{2} = k'(k/2)(T - k'T). \tag{25}
$$

Adding inequalities (24) and (25), and dividing the resulting inequality by k' yields the z-clique inequality (23).

Second, suppose that $R = kk' - 1$, i.e., that $|C| = kk'T + (kk' - 1)$ for some positive integer T. In this case, the z-clique inequality takes the form:

$$
z(C) \geqslant \binom{T+1}{2}(kk'-1) + \binom{T}{2} = k'(k/2)(T+1)T - T. \tag{26}
$$

The y-clique inequality on C takes the form:

$$y(C) \geqslant \binom{Tk'+k'}{2}(k-1) + \binom{Tk'+k'-1}{2}$$
$$= k'(k/2)(T+1)(Tk'+k'-1) - (Tk'+k'-1). \tag{27}$$

and the (y, z)-clique inequality on C can be written as follows:

$$k'z(C) - y(C) \geqslant -(Tk+k-1)\binom{k'}{2} - \binom{k'-1}{2}$$
$$= k'(k/2)(T+1)(1-k') + (k'-1). \tag{28}$$

Adding inequalities (27) and (28), and dividing the resulting inequality by k' yields the z-clique inequality (26). $\qquad\square$

Proof of Theorem 3 It is sufficient to show that for any colouring $C : V \rightarrow \{0, \ldots, kk'-1\}$ there exists a objective-preserving permutation $\sigma : \{0, \ldots, kk'-1\} \rightarrow \{0, \ldots, kk'-1\}$ such that

$$\phi(\sigma \circ C(v)) > v \quad \text{for } v = 1, \ldots, n. \tag{29}$$

By objective-preserving we mean that:

$$c_1 \bmod k = c_2 \bmod k \Longleftrightarrow \sigma(c_1) \bmod k = \sigma(c_2) \bmod k \quad \text{for all } 0 < c_1, c_2 < kk'.$$

We prove that such a permutation exists by induction on the number of nodes in V. For the case $n = 1$, the result holds trivially.

Suppose that the result holds $n \leq l$, and let us now consider the case $n = l+1$. Using our induction hypothesis, we suppose, for notational convenience, that $\phi(C(v)) < v$ for $v = 1, \ldots l$. Now, let $C(l+1) = pk + r$. If $\phi(C(l+1)) = p + r < l + 1$ then the colouring already satisfies the required condition so we assume that this is not true.

We consider two cases. In the first case, suppose that $C(v) \bmod k \neq r$ for all $v = 1, \ldots, l$. This can be split into two further subcases. In the first subcase, we assume that $l + 1 \leqslant k$. We now define an objective-preserving permutation which assigns the colour l to node $l + 1$. Note that by induction hypothesis, none of the nodes $v = 1, \ldots, l$ are assigned to colour l. Define permutations σ_1 and σ_2 as follows:

$$\sigma_1 : \begin{cases} tk + r \mapsto tk + l \\ tk + l \mapsto tk + r & \text{for } t = 0, \ldots, k'-1 \\ c \mapsto c & \text{otherwise,} \end{cases}$$

$$\sigma_2 : \begin{cases} pk + l \mapsto l \\ l \mapsto pk + l \\ c \mapsto c & \text{otherwise,} \end{cases}$$

then $\sigma_2 \circ \sigma_1 \circ \mathcal{C}(l+1) = l$ and so we have $\phi(\sigma_2 \circ \sigma_1 \circ \mathcal{C}(l+1)) = l < l+1$ as required.

In the second subcase we assume that $l+1 > k$. In this case, the following objective-preserving permutation can be used:

$$\sigma : \begin{cases} pk + r \mapsto r \\ r \mapsto pk + r \\ c \mapsto c & \text{otherwise.} \end{cases}$$

Then, $\phi(\sigma \circ \mathcal{C}(l+1)) = r < k < l+1$ as required.

In the second case, we suppose that the set $C_{rl} := \{1 \le v \le l : \mathcal{C}(v) \bmod k = r\}$ is non-empty, and let $u = \max C_{rl}$. Then, for some $0 \le s < p$ we have $\mathcal{C}(u) = sk + r$. We now define the required permutation:

$$\sigma : \begin{cases} pk + r \mapsto (s+1)k + r \\ (s+1)k + r \mapsto pk + r \\ c \mapsto c & \text{otherwise.} \end{cases}$$

Then, $\phi(\sigma \circ \mathcal{C}(l+1)) = (s+1) + r < u + 1 \le l+1$ as required, where the first inequality follows from our induction hypothesis. $\square$

References

1. Aardal, K.I., van Hoesel, S.P.M., Koster, A.M.C.A., Mannino, C., Sassano, A.: Models and solution techniques for frequency assignment problems. Ann. Oper. Res. **153**(1), 79–127 (2007)
2. Ales, Z., Knippel, A., Pauchet, A.: Polyhedral combinatorics of the k-partitioning problem with representative variables. Discrete Appl. Math. **211**, 1–14 (2016)
3. Anjos, M.F., Ghaddar, B., Hupp, L., Liers, F., Wiegele, A.: Solving k-way graph partitioning problems to optimality: the impact of semidefinite relaxations and the bundle method. In: Jünger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 355–386. Springer, Berlin (2013)
4. Borndörfer, R., Weismantel, R.: Set packing relaxations of integer programs. Math. Program. **88**, 425–450 (2000)
5. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973). https://doi.org/10.1145/362342.362367
6. Caprara, A., Fischetti, M.: $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts. Math. Program. **74**, 221–235 (1996)
7. Carlson, R.C., Nemhauser, G.L.: Scheduling to minimize interaction cost. Oper. Res. **14**(1), 52–58 (1966)
8. Chatziafratis, V., Charikar, M.: Approximate hierarchical clustering via sparsest cut and spreading metrics. In: P. Klein (ed.) Proceedings of SODA 2017, to appear. SIAM, Philadelphia, PA (2017)
9. Chopra, S., Rao, M.R.: The partition problem. Math. Program. **59**(1–3), 87–115 (1993)
10. Chopra, S., Rao, M.R.: Facets of the k-partition polytope. Discrete Appl. Math. **61**(1), 27–48 (1995)
11. Christof, T., Löbel, A., Stoer, M.: PORTA—a polyhedron representation transformation algorithm. Software package, available for download at http://www.zib.de/Optimization/Software/Porta (1997)
12. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal Complex Syst. **1695** (2006). http://igraph.org
13. Dasgupta, S.: A cost function for similarity-based hierarchical clustering. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing, pp. 118–127. ACM (2016)
14. de Sousa, V.J.R., Anjos, M.F., Le Digabel, S.: Computational study of valid inequalities for the maximum k-cut problem. Working paper, available on optimization online (2016)

15. Deza, M., Grötschel, M., Laurent, M.: Complete descriptions of small multicut polytopes. In: Gritzmann, P., Sturmfels, B. (eds.) Applied Geometry and Discrete Mathematics. AMS, Philadelphia (1990)
16. Deza, M., Grötschel, M., Laurent, M.: Clique-web facets for multicut polytopes. Math. Oper. Res. **17**(4), 981–1000 (1992)
17. Eisenblätter, A.: Frequency assignment in GSM networks: Models, heuristics, and lower bounds. Ph.D. thesis, Technical University of Berlin (2001)
18. Eisenblätter, A.: The semidefinite relaxation of the k-partition polytope is strong. In: W.J. Cook, A.S. Schulz (eds.) Proceedings of IPCO IX, pp. 273–290. Springer, Berlin (2002)
19. Eppstein, D., Löffler, M., Strash, D.: Listing all maximal cliques in sparse graphs in near-optimal time. In: Cheong, O., Chwa, K., Park, K. (eds.) Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15–17, 2010, Proceedings, Part I, pp. 403–414. Springer, Berlin/Heidelberg (2010)
20. Fairbrother, J., Letchford, A.N.: Projection results for the k-partition problem. Discrete Optim. (2017). https://doi.org/10.1016/j.disopt.2017.08.001. http://www.sciencedirect.com/science/article/pii/S1572528617301718
21. Frieze, A., Jerrum, M.: Improved approximation algorithms for max k-cut and max bisection. Algorithmica **18**(1), 67–81 (1997)
22. Ghaddar, B., Anjos, M.F., Liers, F.: A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. Ann. Oper. Res. **188**(1), 155–174 (2011)
23. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988)
24. Grötschel, M., Wakabayashi, Y.: A cutting plane algorithm for a clustering problem. Math. Program. **45**(1–3), 59–96 (1989)
25. Grötschel, M., Wakabayashi, Y.: Facets of the clique partitioning polytope. Math. Program. **47**(1–3), 367–387 (1990)
26. Gurobi Optimization, I.: Gurobi optimizer reference manual (2016). http://www.gurobi.com
27. Hochbaum, D.S.: Why should biconnected components be identified first [sic]. Discrete Appl. Math. **42**(2–3), 203–210 (1993)
28. Hopcroft, J., Tarjan, R.: Algorithm 447: efficient algorithms for graph manipulation. Commun. ACM **16**(6), 372–378 (1973)
29. Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. Discrete Optim. **8**, 595–610 (2011)
30. Letchford, A.N.: On disjunctive cuts for combinatorial optimization. J. Comb. Optim. **5**, 299–315 (2001)
31. Lu, G., Zhou, M.T., Niu, X.Z., She, K., Tang, Y., Qin, K.: A survey of proximity graphs in wireless networks. J. Softw. **19**(4), 888–911 (2010)
32. Mannino, C., Rossi, F., Rossi, F., Smriglio, S.: A unified view in planning broadcasting networks. In: D. Kurlander, M. Brown, R. Rao (eds.) Proceedings of INOC 2007, pp. 41–50 (2007)
33. Margot, F.: Symmetry in integer linear programming. In: Jünger, M., et al. (eds.) 50 Years of Integer Programming 1958–2008, pp. 647–686. Springer, Berlin (2010)
34. Moon, J.W., Moser, L.: On cliques in graphs. Israel J. Math. **3**(1), 23–28 (1965). https://doi.org/10.1007/BF02760024
35. Müller, R., Schulz, A.S.: Transitive packing: a unifying concept in combinatorial optimization. SIAM J. Optim. **13**, 335–367 (2002)
36. Oosten, M., Rutten, J.H.G.C., Spieksma, F.C.R.: The clique partitioning problem: facets and patching facets. Networks **38**, 209–226 (2009)
37. Rendl, F.: Semidefinite relaxations for partitioning, assignment and ordering problems. 4OR **10**(4), 321–346 (2012)
38. Resende, M., Pardalos, P. (eds.): Handbook of Optimization in Telecommunications. Springer, New York (2007)
39. Roy, A., Pokutta, S.: Hierarchical clustering via spreading metrics. In: Advances in Neural Information Processing Systems, pp. 2316–2324 (2016)
40. Sanders, P., Schulz, C.: Engineering multilevel graph partitioning algorithms. In: Demetrescu, C., Halldórsson, M.M. (eds.) Proceedings of ESA 2011, Lecture Notes in Computer Science, vol. **6942**. Springer, Heidelberg (2011)
41. Seidman, S.B.: Network structure and minimum degree. Soc. Netw. **5**(3), 269–287 (1983)

42. Sotirov, R.: An efficient semidefinite programming relaxation for the graph partition problem. INFORMS J. Comput. **26**(1), 16–30 (2013)
43. Szekeres, G., Wilf, H.S.: An inequality for the chromatic number of a graph. J. Comb. Theory **4**(1), 1–3 (1968)