# Social Network Analysis and Community Detection by Decomposing a Graph into Relaxed Cliques

Timo Gschwind*,[a], Stefan Irnich[a], Fabio Furini[b], Roberto Wolfler Calvo[c]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*
[b] *LAMSADE, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny, 75016 Paris, France.*
[c] *LIPN, Université Paris 13, 93430 Villetaneuse, France.*

**Abstract**

In social network analysis (SNA), relationships between members of a network are encoded in an undirected graph where vertices represent the members of the network and edges indicate the existence of a relationship. One important task in SNA is community detection, that is, clustering the members into communities such that relatively few edges are in the cutsets but relatively many are internal edges. The clustering is intended to reveal hidden or reproduce known features of the network, while the structure of communities is arbitrary. We propose decomposing a graph into the minimum number of relaxed cliques as a new method for community detection especially conceived for cases in which the internal structure of the community is important. Cliques, that is, subgraphs with pairwise connected vertices, can model perfectly cohesive communities, but often they are overly restrictive because many real communities form dense but not complete subgraphs. Therefore, different variants of relaxed cliques have been defined in terms of vertex degree and distance, edge density, and connectivity. They allow to impose application-specific constraints a community has to fulfill such as familiarity and reachability among members and robustness of the communities. Standard compact formulations fail in finding optimal solutions even for small instances of such decomposition problems. Hence, we develop exact algorithms based on Dantzig-Wolfe reformulation and branch-and-price techniques. Extensive computational results demonstrate the effectiveness of all components of the algorithms and the validity of our approach when applied to social network instances from the literature.

*Key words:* Graph decomposition, community detection, clique relaxations, social network analysis, branch-and-price

## 1. Introduction

In *social network analysis* (SNA, Wasserman and Faust, 1994; Scott, 2012) there is a growing interest in studying social networks aiming at extracting knowledge from the herein identified structures and characteristic numbers. The analysis of *cohesive groups* also known as *communities* (or *clusters*, *modules*, *blocks*) has received a lot of attention from researchers of different areas like social and computer science, biology, economics, physics, and discrete mathematics. Classically, cliques have been used to model cohesive groups. They can be seen as extremal cohesive groups in the sense that every member is fully connected with each other. This constraint has been found too restrictive in many applications and, therefore, various relaxations of the clique concept, such as $s$-clique, $s$-plex, $s$-club, $s$-defective clique, and $\gamma$-quasi-clique, have been introduced (see Pattillo *et al.*, 2013a, and references given there). We refer to these structures as *relaxed cliques* in the following.

*Corresponding author.
*Email addresses:* gschwind@uni-mainz.de (Timo Gschwind), irnich@uni-mainz.de (Stefan Irnich), fabio.furini@dauphine.fr (Fabio Furini), roberto.wolfler@lipn.univ-paris13.fr (Roberto Wolfler Calvo)

To the best of our knowledge, the graph-theory and OR literature on clique relaxations has almost exclusively studied questions related to identifying their structural properties or to solve optimization problems in which a maximal or maximum relaxed clique has to be determined. Surprisingly, no research has addressed the related problems of partitioning or covering a graph by the smallest number of relaxed cliques, although this type of question was coined by Balasundaram *et al.* (2011, p. 141).

Decomposing a graph into the minimal number of cliques is well-known as the clique partitioning problem. It is equivalent to finding a minimum vertex coloring for the complement graph, and has been discussed intensively, e.g., by Mehrotra and Trick (1998); Nemhauser and Park (1991), and Held *et al.* (2012). We extend this stream of research and propose decomposing a graph into the minimum number of relaxed cliques as a new method for community detection.

We study the cases of partitioning (disjoint clusters) as well as covering (overlapping clusters). Any subgraph of a clique is again a clique, a property known as *hereditary* (Yannakakis, 1978). However, for some classes of relaxed cliques, this is not generally true. As a consequence, partitioning and covering a graph with a minimum number of relaxed cliques can be a different problem, i.e., covering can be a proper relaxation of partitioning.

The proposed method is applicable in those cases in which one has a good understanding of the structural properties that a community must have. Aspects such as *familiarity* among members (few strangers), *reachability* among members (quick communication), and *robustness* of the subgroup (not easily destroyable) are often desirable properties of a community (Balasundaram *et al.*, 2011). In a graph-theoretic description, familiarity concerns vertex degrees, reachability concerns distances, and robustness concerns connectivity. Clique relaxations like $k$-core/$s$-plex, $s$-club/clique, and $k$-block/$s$-bundle, respectively, can model such desired characteristics of subgroups in SNA. Moreover, input data describing a social network may stem from sources that contains errors. In that case using $s$-defective cliques or $\gamma$-quasi-cliques offers a way of absorbing such inaccuracies.

### *1.1. Literature Review*

The idea of decomposing a graph $G = (V, E)$ into partitions is not new. We briefly review related research fields such as graph partitioning, community detection, and graph clustering in order to point out the similarities and differences to the new type of problems introduced within the paper at hand.

In *graph partitioning*, the task is to find a partitioning of the vertex set $V$ into $p$ blocks $V_1, V_2, \ldots, V_p$. Typically, a weight is associated to each vertex of the graph and a maximum capacity of each partition must be respected. We refer to (Garey and Johnson, 1979) where the *graph partitioning problem* has been formally defined and to (Buluç *et al.*, 2013) for a recent comprehensive overview. In addition, balancing constraints requiring that partitions are of (almost) equal size or weight can be considered. If weights are associated to the edges of the graph, the objective can be to minimize the weight of the edge cutsets $E(V_i, V_j)$ comprising all edges connecting different partitions $V_i$ and $V_j$ with $i < j$. Applications of graph partitioning are widespread and include, e.g., the distribution of work in parallel processing, image processing, sparse matrix factorization, very large-scale integration (VLSI) design, and the pre-computation of information to accelerate shortest-path queries in routing. The focus in graph partitioning is on the relationship between different clusters, while the structure of a cluster is almost irrelevant except for its size or weight. In contrast, the primary focus of our work is on the identification of clusters that have very specific structural properties, i.e., they are relaxed cliques of a specific type.

*Community detection* is strongly related to our work. The articles by Porter *et al.* (2009) and Schaeffer (2007) and the survey by Fortunato (2010) provide comprehensive overviews showing that diverse methods from various fields have found their way into community detection. Informally speaking, community detection is concerned with clustering the vertex set $V$ of a graph $G = (V, E)$ into communities $V_1, V_2, \ldots, V_p$ such that relatively few edges are found in the cutsets $E(V_i, V_j)$ but relatively many are internal edges $E(V_i)$. In particular, the number $p$ of communities within a given network is typically unknown. In contrast to the balancing constraints in graph partitioning, the communities are generally of unequal size, moreover, their density and structure (if any) can vary. Without explicitly formalizing the relaxed clique partitioning and covering problem, Fortunato (2010) and others offer the concept of relaxed cliques as a viable and reasonable

concept for identifying communities. However, methods for clustering a network into relaxed cliques in a "best possible way" are not described there. This is probably because the standard objectives in community detection asses a clustering by considering both internal edges and edges in the cutsets of the clusters. A very common quality measure in this context is *modularity*, originally introduced by Newman and Girvan (2004): Given $p$ clusters $V_1, V_2, \ldots, V_p$, the *modularity* is $Q(V_1, V_2, \ldots, V_p) = \sum_{i=1}^{p} \left( \frac{|E(V_i)|}{|E|} - exp(V_i) \right)$, where $exp(V_i)$ is the expected fraction of inner-cluster edges of $V_i$. A common assumption on the underlying distribution (the so-called *null model*) is that an edge between two vertices $i$ and $j$ appears with probability $|N(i)||N(j)|/|E|$, where $N(i)$ and $N(j)$ is the set of neighboring vertices of $i$ and $j$, respectively. The modularity value $Q(V_1, V_2, \ldots, V_p)$ varies between 0 and 1 depending on the clustering. A value close to 1 indicates that a strong community structure has been identified. Indeed, modularity maximization seems to be appropriate as a general tool for identifying communities with a non-specified structure. Aloise *et al.* (2010) propose new column-generation algorithms to exactly solve the modularity maximization problem in networks. They compare a row-generation algorithm and a direct solution approach using CPLEX to solve a 0-1 MIQP with three column-generation algorithms differing in the underlying formulation and algorithm to solve the respective pricing problem. One result is that the column-generation approach with a sparse quadratic subproblem formulation outperforms the other algorithms, while the question of finally producing integer solutions via branching is not addressed in the article.

*Clustering methods* require a measure of distance (or similarity). In the context of graphs, it means that vertices are considered as (data) points in a metric space. The goal is to find a partitioning of the points $V$ into $p$ clusters ($p$ is typically given) and to minimize or maximize an objective based on distances between points and/or from points to cluster centroids (Fortunato, 2010, p. 93f). Examples are $p$-means clustering where the average squared distance between points and centroids is minimized, $p$-centroid where the maximum distance between points and centroids is minimized, and $p$-clustering sum where the sum of all intra-cluster distances is minimized. While traditional clustering methods need $p$ and additional attributes of vertices as inputs (data points), we solely rely on adjacency information as an input. It is also possible to define a distance by pure adjacency-based measures (e.g. Schaeffer, 2007, p. 36) such as the overlap $|N(i) \cap N(j)|/|N(i) \cup N(j)|$. However, there are no well-defined structural requirements for sharply distinguishing between feasible and infeasible clusters.

Finally, the work of Guo *et al.* (2010) has some relation to the problems studied here, but with a completely different objective and context. The authors consider the so-called *s-plex cluster editing problem* in which, for an undirected graph $G = (V, E)$ and an integer $p \geq 0$, the question is whether $G$ can be modified by up to $p$ edge deletions and insertions into a graph whose connected components are $s$-plexes.

### *1.2. Paper Contribution*

The contributions of the paper at hand are the following:

- the formal introduction of various relaxed clique partitioning and covering problems as a new approach for community detection;
- the development of branch-and-price algorithms for their exact solution;
- the introduction of connectivity conditions that each relaxed clique has to respect and the discussion of their implications on pricing algorithms and branching;
- two new mathematical formulations for finding $k$-blocks and $s$-bundles, respectively;
- new combinatorial branch-and-bound algorithms for finding maximum weight $s$-clubs and connected hereditary relaxed cliques able to cope with general weights;
- the presentation of branching rules that are structure preserving in the sense that the pricing problem remains structurally unchanged;
- the proof that covering and partitioning a graph with connected $s$-cliques is equivalent;

- the presentation of a comprehensive computational study including the application of the new models and algorithms for detecting communities in some real-world social networks that are intensively studied in the SNA and community-detection literature.

The remainder of this paper is organized as follows: In Section 2, we provide an overview over the first-order clique relaxations. In Section 3, we present and discuss a generic formulation for finding a maximum cardinality or maximum weight relaxed clique in a graph. In Section 4, we derive a mixed integer programming (MIP) formulation for the partitioning and covering problem, which is compact whenever the formulation for finding a relaxed clique is compact. The branch-and-price solution algorithm is explained in Section 5: Besides the presentation of the column-generation master and pricing problem, the heart of this section is the development and discussion of possible branching schemes for both the partitioning and covering case. Computational results are presented and discussed in Section 6. Final conclusions and an outlook in Section 7 close the paper.

## 2. Clique Relaxations

In this section, we introduce the basic notation and different types of relaxed cliques following the taxonomy offered by Pattillo *et al.* (2013a). From now on, we assume that a simple graph $G = (V, E)$ with finite vertex set $V$ and edge set $E$ is given. For any subset $S \subseteq V$, the vertex-induced subgraph of $S$ is $G[S] = (S, E \cap (S \times S))$.

In the following, $i \in V$ is any vertex and $S \subseteq V$ is any vertex set. Vertices adjacent to $i$ are denoted by $N(i)$. A set $S$ is a *clique* if $G[S]$ is complete, i.e., all vertices are adjacent. Cliques $S$ form extreme subsets, since all vertices have maximum degree $|S| - 1$, the distance between any two vertices is 1, $G[S]$ has maximum density of 1, and is $(|S| - 1)$-connected.

### *2.1. Definitions of Relaxed Cliques*

The following relaxed cliques are obtained by relaxing a single aspect of the clique definition. In the literature, they are referred to as first-order clique relaxations, while the clique itself is named zero-order clique relaxation (Pattillo *et al.*, 2013a, p. 12).

*Relaxing Degree..* The *vertex degree* of $i$ is $|N(i)|$ and is denoted by $\deg_G(i)$. The *minimum vertex degree* of $G$ is $\delta(G) = \min_{i \in V} \deg(i)$. For $k \geq 0$, $S$ is a *$k$-core* if $\delta(G[S]) \geq k$. For $s \geq 1$, $S$ is an *$s$-plex* if $\delta(G[S]) \geq |S| - s$. Every $s$-plex is an $(|S| - s)$-core, and vice versa. The $s$-plex clique relaxation has been studied, e.g., in the context of transmission network analysis in Tuberculosis contact investigations by Cook *et al.* (2007).

*Relaxing Distance..* For two vertices $i, j \in V$, $\text{dist}_G(i, j)$ is the minimum *distance* between $i$ and $j$, i.e., the minimum length of an $i$-$j$-path in $G$. Note that the length of a path is given by the number of its edges and that $\text{dist}_G(i, j) = \infty$ if $i$ and $j$ are disconnected in $G$. For $s \geq 1$, $S$ is an *$s$-clique* if $\text{dist}_G(i, j) \leq s$ for all $i, j \in S$. An $s$-clique is an ordinary clique (1-clique) in the $s$th power graph $G^s = (V, \{\{i, j\} : i, j \in V, i < j, d_G(i, j) \leq s\})$, and vice versa. The maximum distance is the diameter of $G$ given by $\text{diam}(G) = \max_{i \neq j} \text{dist}_G(i, j)$. For $s \geq 1$, $S$ is an *$s$-club* if $\text{dist}_{G[S]}(i, j) \leq s$ for all $i, j \in S$ or equivalent $\text{diam}(G[S]) \leq s$. Any $s$-club is an $s$-clique, but the reverse it not necessarily true. The $s$-club and $s$-clique relaxations have been intensively studied and their relevance for network optimization applications in biology were pointed out by Almeida and Carvalho (2012).

*Relaxing Density..* For any $S \subseteq V$, the edge set $E(S)$ is the set of edges in $G$ with both endpoints in $S$. Moreover, the *edge density* of a subgraph $G[S]$ is defined as $\rho(G[S]) = |E(S)| / \binom{|S|}{2}$. For $0 \leq \gamma \leq 1$, $S$ is a *$\gamma$-quasi-clique* if $\rho(G[S]) \geq \gamma$. While the density is a relative measure for existing/missing edges, one can also count their number. For $s \geq 0$, $S$ is an *$s$-defective clique* if $|E(S)| \geq \binom{|S|}{2} - s$. Hence, any $s$-defective clique has a density of at least $\gamma = 1 - s / \binom{|S|}{2}$, i.e., is a $\gamma$-quasi-clique, and vice versa. The $s$-defective clique has been used, e.g., to identify large protein interaction networks using noisy data collected from large-scale (high-throughput) experiments (Yu *et al.*, 2006).

*Relaxing Connectivity..* A set $C \subset V$ is a *vertex cut* of a connected graph $G = (V, E)$ if $G[V \setminus C]$ is a disconnected graph. Note that any vertex cut $C$ has at most $|V| - 2$ elements. The vertex connectivity $\kappa(G)$ is the size of a minimum vertex cut. For cliques $S$, $G[S]$ does not have any vertex cuts, and therefore one defines $\kappa(G[S]) = |S| - 1$. A graph is called $k$-vertex-connected if its vertex connectivity is $k$ or greater. Let $i, j \in V$ be two different, non-adjacent vertices. The local connectivity $\kappa_G(i, j)$ is the minimum size of a vertex cut $C$ disconnecting $i$ and $j$ in $G[V \setminus C]$. For adjacent vertices $i$ and $j$, one defines $\kappa_G(i, j) = \infty$. Then, if $G$ is not a clique, $\kappa(G)$ equals the minimum of $\kappa_G(i, j)$ over all pairs of different vertices $i, j \in V$.

Two $i$-$j$-paths are called *vertex-disjoint* if they have no vertices in common except $i$ and $j$. According to Menger's theorem (Menger, 1927), the minimum size of a vertex cut disconnecting $i$ and $j$ is the maximum number of vertex-disjoint paths connecting $i$ and $j$. Therefore, for non-adjacent vertices $i$ and $j$, $\kappa_G(i, j)$ is the maximum number of vertex-disjoint $i$-$j$-paths. For $k \geq 1$, $S$ is a *$k$-block* if $\kappa(G[S]) \geq k$. For $s \geq 1$, $S$ is an *$s$-bundle* if $\kappa(G[S]) \geq |S| - s$. By definition, singleton sets $S = \{i\}$ are no $k$-blocks but always $s$-bundles. Connectivity and $k$-blocks have been comprehensively surveyed by Kammer and Täubig (2005). To the best of our knowledge, the $s$-bundle relaxation coined in (Pattillo *et al.*, 2013a) has only been studied in (Gschwind *et al.*, 2015).

Table 1 summarizes the definitions of the eight first-order relaxed cliques. In *higher-order clique relaxations*, more than one aspect of the clique definition is relaxed. For example, the ($\lambda$, $\gamma$)-quasi-clique is a second-order relaxation relaxing degree and density so that each vertex must be connected to at least $\lambda(|S| - 1)$ vertices and the induced subgraph must have a density not smaller than $\gamma$. Note that in some cases one property may already result from another property. For an overview of dependencies between first-order relaxations see (Pattillo *et al.*, 2013a, Table 2).

Table 1: Definition of different clique relaxations

| Type of relaxation | Definition | Based on | Clique | Hereditary | Connected |
|---|---|---|---|---|---|
| $k$-core | $\delta(G[S]) \geq k$ | Degree | $k = \lvert S\rvert - 1$ | no | $\lvert S\rvert \leq 2k + 1$ |
| $s$-plex | $\delta(G[S]) \geq \lvert S\rvert - s$ | Degree | $s = 1$ | yes | $\lvert S\rvert \geq 2s - 1$ |
| $s$-clique | $\mathrm{dist}_G(i, j) \leq s$ for all $i, j \in S$ | Distance | $s = 1$ | yes | $s = 1$ |
| $s$-club | $\mathrm{diam}(G[S]) \leq s$ | Distance | $s = 1$ | no | always |
| $\gamma$-quasi-clique | $\rho(G[S]) \geq \gamma$ | Density | $\gamma = 1$ | no | $\left\lceil \gamma\binom{\lvert S\rvert}{2} - \binom{\lvert S\rvert - 1}{2} \right\rceil \geq 1$ |
| $s$-defective clique | $\lvert E(G[S])\rvert \geq \binom{\lvert S\rvert}{2} - s$ | Density | $s = 0$ | yes | $\lvert S\rvert \geq s + 2$ |
| $k$-block | $\kappa(G[S]) \geq k$ | Connectivity | $k = \lvert S\rvert - 1$ | no | always |
| $s$-bundle | $\kappa(G[S]) \geq \lvert S\rvert - s$ | Connectivity | $s = 1$ | yes | $\lvert S\rvert \geq s + 1$ |

Note: The last column gives sufficient conditions for connectivity (Pattillo *et al.*, 2013a, p. 17).

*Requiring Connectivity..* In many practical applications, clusters need to be connected. For community detection, e.g., Fortunato (2010, p. 84) stresses that connectedness is a required property. If a community were disconnected, it could be considered as two or more smaller groups. A weakness of general relaxed cliques is that they are not necessarily connected, see last column of Table 1. Indeed, arbitrarily large $s$-cliques can be disconnected because the removal of the central vertex from a star graph induces an edgeless graph, which is however a 2-clique and therefore also an $s$-clique for all $s \geq 2$. Also, arbitrarily large disconnected $\gamma$-quasi-cliques exist resulting from the addition of an isolated vertex to a clique. In contrast, this phenomenon occurs only for small-sized $S \subset V$ in case of $s$-plex, $s$-defective clique, and $s$-bundle, see Figure 1.

As a consequence, we suggest to consider *connected relaxed cliques* $S$ as feasible structures, which result from requiring connectivity of the induced subgraph $G[S]$ in addition to the definition of the respective relaxed clique. Note that for hereditary relaxed cliques ($s$-plex, $s$-clique, $s$-defective clique, and $s$-bundle) the connectivity requirement makes the resulting structures non-hereditary. For example, a path with three vertices forms a connected 1-defective clique, which becomes disconnected when the middle vertex is

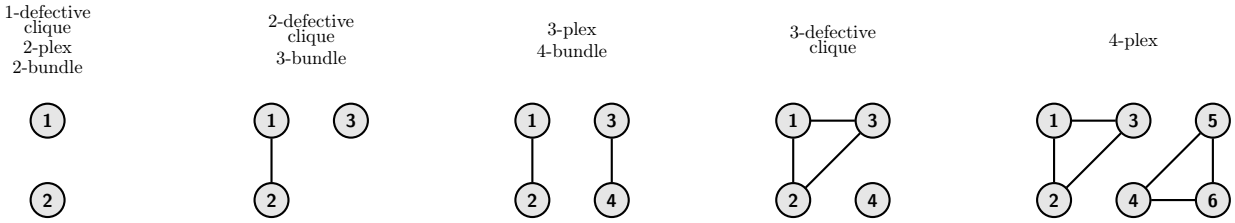


Figure 1: Largest disconnected $s$-plex, $s$-defective clique, and $s$-bundle

removed. This has important consequences for the applicability of existing algorithms, and we discuss this issue in Section 5.1.

*2.2. Related Optimization Problems*

The scientific literature has focused mainly on finding relaxed cliques that are large. The attribute large may refer to relaxed cliques $S$ that either are of maximum cardinality, are maximal with respect to inclusion, or have maximum weight.

*Maximum Cardinality Relaxed Cliques..* Before we define the related optimization problems, it is helpful to describe some properties in order to classify types of relaxed cliques. Let $\Pi$ be the *graph property*, e.g., describing a specific clique relaxation. According to Yannakakis (1978), a property $\Pi$ is *nontrivial* if it is true for all graphs $G[S]$ induced by singleton sets $S = \{i\}$, but not fulfilled for every graph. $\Pi$ is *interesting* if there exist arbitrarily large graphs satisfying $\Pi$. Moreover, it is *hereditary on induced subgraphs* if for any $S \subseteq V$ with $G[S]$ has property $\Pi$ it follows that also $G[S']$ has property $\Pi$ for any $S' \subset S, S' \neq \emptyset$.

The problem of finding a relaxed clique $S \subseteq V$ with largest cardinality $|S|$ is known as the *maximum (cardinality) relaxed clique problem* (MC-RC). For nontrivial, interesting, and hereditary properties $\Pi$, Yannakakis (1978) has shown that MC-RC is $\mathcal{NP}$-hard. It is straightforward to see that $\Pi$ is hereditary for $s$-plex, $s$-clique, $s$-defective clique, and $s$-bundle. Consequently, MC-RC is an $\mathcal{NP}$-hard problem for this type of relaxed cliques.

The properties $\Pi$ of being a $k$-core, $s$-club (for $s > 1$), $\gamma$-quasi-clique (for $\gamma < 1$), or $k$-block are not hereditary. The theorem by Yannakakis (1978) is therefore not applicable. Indeed, the maximum cardinality $k$-core problem is polynomially solvable (see Kosub, 2004). The computation of the $k$-connected components of a graph (herewith also solving the $k$-block partitioning problem) can be done in polynomial time for fixed $k$ (see Kammer and Täubig, 2005). For $s$-club, the $\mathcal{NP}$-hardness of MC-RC was proven by (Bourjolly *et al.*, 2002). Recently, Pattillo *et al.* (2013b) showed that MC-RC for $\gamma$-quasi-cliques is $\mathcal{NP}$-complete.

Table 2 summarizes the exact solution approaches for MC-RC for the first-order clique relaxations. Exact algorithms for clique are too numerous to be listed here and we refer to (Carraghan and Pardalos, 1990; Östergård, 2002) and the survey (Abello *et al.*, 1999). Note that these algorithms can solve MC-RC for $s$-cliques by considering the $s$th power graph.

*Inclusion Maximal Relaxed Cliques..* If a subset $S \subseteq V$ is a largest relaxed clique with respect to inclusion then $S$ is a *maximal relaxed clique*. Obviously, any maximum relaxed clique is also maximal, but the reverse is not necessarily true. For all variants, the question whether or not $S$ induces a relaxed clique is efficiently decidable. Therefore, for hereditary $\Pi$, finding inclusion maximal relaxed cliques can be done efficiently by adding vertices in a one-by-one fashion. In contrast, the maximality test regarding subset inclusion is $\mathcal{NP}$-hard for $s$-club as shown by Mahdavi Pajouh and Balasundaram (2012).

*Maximum Weight Relaxed Cliques..* If weights $w_i \in \mathbb{R}$ are given for all vertices $i \in V$, the *maximum weight relaxed clique* problem (MW-RC) consists of finding a subset $S \subseteq V$ such that $w(S) = \sum_{i \in S} w_i$ is maximum and $S$ is a relaxed clique. Clearly, with unit weights MW-RC reduces to MC-RC. For relaxed cliques with hereditary $\Pi$, it is no restriction to assume that weights $w_i$ are non-negative because otherwise a vertex with negative weight can be eliminated from the consideration. For non-hereditary $\Pi$, vertices with negative weight need to be considered as discussed in Section 5.1.

Table 2: Exact algorithms for MC-RC for different clique relaxations

| Clique relaxation | Exact approach and reference |
|---|---|
| $k$-core | polynom. solvable, see (Kosub, 2004) |
| $s$-plex | B&C:(Balasundaram *et al.*, 2011), CB&B:(Trukhanov *et al.*, 2013; Gschwind *et al.*, 2015) |
| $s$-clique | (any algorithm for clique) |
| $s$-club | B&C:(Almeida and Carvalho, 2012, 2013), CB&B:(Bourjolly *et al.*, 2002; Mahdavi Pajouh and Balasundaram, 2012; Shahinpour and Butenko, 2013), MIP:(Bourjolly *et al.*, 2000; Veremyev and Boginski, 2012), SAT:(Wotzlaw, 2014) |
| $\gamma$-quasi-clique | MIP:(Pattillo *et al.*, 2013b) |
| $s$-defective clique | B&C:(Sherali and Smith, 2006), CB&B:(Trukhanov *et al.*, 2013; Gschwind *et al.*, 2015) |
| $k$-block | polynom. solvable, see (Kammer and Täubig, 2005) |
| $s$-bundle | CB&B:(Gschwind *et al.*, 2015) |

B&C=branch-and-cut, CB&B=combinatorial branch-and-bound, MIP=(mixed) integer model (no cutting planes), SAT=formulation as a partial max-sat problem

## 3. Mathematical Formulations for Relaxed Cliques

Different formulations for the MC-RC and MW-RC variants have been suggested in the literature (see Pattillo *et al.*, 2012, 2013a, for an overview). All formulations use either variables $x_i \in \{0,1\}$ to indicate that vertex $i \in V$ is in the relaxed clique $S$, or variables $y_e \in \{0,1\}$ to indicate that $G[S]$ contains edge $e \in E$, or both. The properties defining $\Pi$ can be formulated using MIP. The relaxed cliques can be described with the help of a polytope describing a set $\mathscr{F}(G)$ of integer points such that $(\mathbf{x},\mathbf{y}) \in \mathscr{F}(G)$ holds if and only if $G[S]$ with $S = \{i \in V : x_i = 1\}$ fulfills $\Pi$. We can write the following generic model for MW-RC:

$$\max \sum_{i \in V} w_i x_i, \quad \text{s.t.} \quad (\mathbf{x},\mathbf{y}) \in \mathscr{F}(G) \tag{1}$$

A possible way to ensure the compatibility of vertex and edge variables is setting $x_i x_j = y_{ij}$ for all $\{i,j\} \in E$ and to apply the McCormick (1976) linearization for binary variables. We assume that this or any alternative coupling mechanism is already part of the definition of $\mathscr{F}(G)$. Note that additional variables, other than $x$ and $y$, may be used to define the set $\mathscr{F}(G)$ or that in some formulations the $y$ variables are useless.

Several mathematical formulation describing the first-order relaxed cliques introduced in Section 2 can be found in the literature. An ordinary clique is described by $\mathscr{F}(G) = \{x_i \in \{0,1\} : x_i + x_j \leq 1 \text{ for all } i,j \in V, i < j \text{ with } \{i,j\} \notin E\}$. Polyhedral results can be found in (Nemhauser and Trotter Jr., 1974, 1975). These results transfer directly to $s$-cliques with $s \geq 2$, since an $s$-clique is an ordinary clique in the power graph $G^s$.

*$s$-Plex..* Balasundaram *et al.* (2011) provide the following compact formulation for $s$-plex. Here, $\mathscr{F}(G) = \{x_i \in \{0,1\} : \sum_{j \in V \setminus N(i)} x_j \leq (s-1)x_i + \bar{d}_i(1-x_i) \text{ for all } i \in V\}$, where the constant $\bar{d}_i$ is defined as $|V \setminus N(i)| - 1$.

*$s$-Defective Clique..* The complement of an $s$-defective clique is a *generalized vertex packing* (GVP-$s$, Sherali and Smith, 2006). Therefore, $s$-defective cliques can be detected as GVP-$s$ in the complement graph $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{\{i,j\} : i,j \in V, i < j, \{i,j\} \notin E\}$. The set $\mathscr{F}(G)$ is given by $\{x_i \in \{0,1\}, \bar{y}_{ij} \geq 0 : \bar{y}_{ij} \geq x_i + x_j - 1, \{i,j\} \in \bar{E}; \sum_{\{i,j\} \in \bar{E}} \bar{y}_{ij} \leq k\}$. It has additional $\bar{y}_{ij}$ variables for all $\{i,j\} \in \bar{E}$.

*$\gamma$-Quasi-Clique..* Pattillo *et al.* (2013b) describe $\gamma$-quasi-cliques by $\mathscr{F}(G) = \{x_i \in \{0,1\}, y_{ij} \geq 0 : \sum_{i<j} (\gamma - a_{ij}) y_{ij} \leq 0; y_{ij} \leq x_i, y_{ij} \leq x_j, y_{ij} \geq x_i + x_j - 1 \text{ for } i,j \in V, i < j\}$, where $(a_{ij})$ is the adjacency matrix of $G$ and the $y_{ij}$ variables are defined for every pair of vertices $i,j \in V, i < j$. The authors also present a more compact formulation with $|V|$ binary and $|V|$ continuous variables and $4|V| + 1$ constraints.

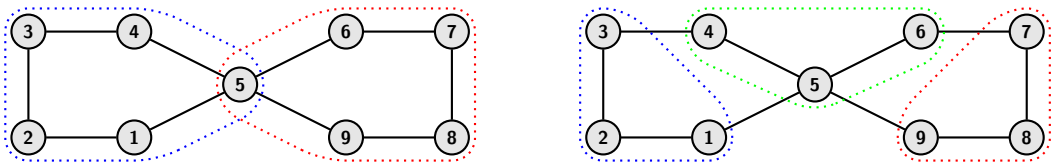


Figure 2: Covering and partitioning into a minimum number of 2-clubs.

*s-Club.*. Several formulations for maximum cardinality $s$-club are known. The path-based formulation by Bourjolly *et al.* (2000) uses indicator variables $x_i$ for the vertices and additional variables for all paths of length at most $s$. With the coupling of both types of variables, the number of variables and constraints is bounded by the number of paths, which is of the order of $\mathcal{O}(|V|^{s+1})$ for dense graphs. However, for $s = 2$ and $s = 3$ the formulation is compact, i.e., polynomial in $|V|$ and $|E|$ and valid inequalities together with a branch-and-cut algorithm were presented by Carvalho and Almeida (2011); Almeida and Carvalho (2012). Veremyev and Boginski (2012) proposed the first compact formulation with a polynomial number of variables and constraints (polynomial in $s$, $|V|$, and $|E|$). Since this formulation is relatively spacious, we describe it in Section A of the Appendix.

To the best of our knowledge, no MIP formulations for $k$-block and $s$-bundle have been presented in the literature. We suggest formulations in Section B of the Appendix.

As discussed above, some types of relaxed cliques are not necessarily connected. In the presented MIP formulations, a straightforward way to impose connectivity is to add constraints

$$\sum_{i \in S} x_i + \sum_{i \in V \setminus S} (1 - x_i) \leq |V| - 1 \qquad S \subseteq V : \kappa(G[S]) \geq 2. \tag{2}$$

In general, this is an exponential number of constraints requiring a cutting-plane procedure to solve the MIP.

## 4. Partitioning and Covering a Graph with Relaxed Cliques

Community detection consists in partitioning or covering a graph with clusters. We propose a new approach for community detection based on decomposing the graph into a minimum number of relaxed cliques. No reasonable problem results for $k$-core because some vertices may have a degree smaller than $k$ and cannot belong to any $k$-core. For $k$-block, the resulting problem is to determine the $k$-connected components, for which efficient algorithms exist (Kammer and Täubig, 2005). Therefore, we restrict ourselves to the six remaining first-order clique relaxations.

According to Porter *et al.* (2009) the 'detection of network communities that overlap is especially appealing in the social sciences, as people belong simultaneously to several communities (constructed via colleagues, family, hobbies, etc.)'. Clearly, covering is always a relaxation of partitioning and this relaxation is proper for non-hereditary structures. The decomposition into 2-clubs shown in Figure 2 is an example. The non-heredity of a particular structure may either result from the property $\Pi$ defining the type of relaxed clique or from the connectivity requirement.

When connectivity is not already ensured by the definition of the relaxed clique, the variants double and we analyze variants with and without connectivity requirement.

All interesting variants of partitioning and covering with first-order relaxed cliques are summarized in Table 3. Since partitioning and covering are identical for hereditary $\Pi$ and without connectivity requirement, $s$-plex, $s$-defective clique, and $s$-bundle are listed in the partitioning column only. Moreover, for $s \geq 2$ and with connectivity imposed, partitioning and covering with connected $s$-cliques differ from clique partitioning in the power graph $G^s$ or vertex coloring in the complement graph (not considered in this paper here). Figure 3 provides an example.

Finally, the following non-trivial result explains why partitioning and covering with connected $s$-cliques is equivalent, resulting in a single entry for $s$-clique partitioning in Table 3:

Table 3: Variants of partitioning and covering with relaxed cliques

| | Partitioning | Covering | Subproblem algorithm |
|---|---|---|---|
| Connected | $s$-plex | $s$-plex | mRDS, MIP-CP |
| | $s$-clique$^\dagger$ | | mRDS, MIP-CP |
| | $\gamma$-quasi | $\gamma$-quasi | MIP-CP |
| | $s$-defective | $s$-defective | mRDS, MIP-CP |
| | $s$-bundle | $s$-bundle | mRDS, MIP-CP |
| General | $s$-plex | | RDS, MIP |
| | $s$-club | $s$-club | CB&B, MIP |
| | $\gamma$-quasi | $\gamma$-quasi | MIP |
| | $s$-defective | | RDS, MIP |
| | $s$-bundle | | RDS, MIP |

(m)RDS=(modified) Russian Doll Search; MIP(-CP)=Mixed Integer Programming solver (with Cutting Plane algorithm); CB&B=Combinatorial Branch-and-Bound; $^\dagger$ for the equivalence of partitioning and covering see Theorem 1

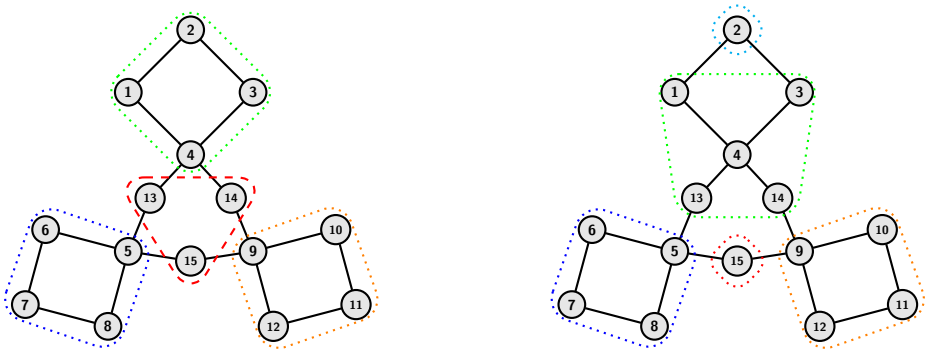


Figure 3: Partitioning into a minimum number of (four) general 2-cliques and (five) connected 2-cliques. Note that $S = \{13, 14, 15\}$ induces the disconnected subgraph $G[S] = (S, \emptyset)$.

**Theorem 1.** *Partitioning and covering a graph with a minimum number of connected $s$-cliques are equivalent problems for all $s \geq 1$. There exists a constructive procedure to transform a covering solution into a partitioning solution using an identical number of $s$-cliques.*

The detailed proof is given in Section F of the Appendix.

A generic compact mathematical formulation for all variants needs an upper bound $\bar{rc}(G)$ on the number of relaxed cliques in a solution so that they can be numbered by $h \in H = \{1, 2, \ldots, \bar{rc}(G)\}$. Then, binary variables $z^h, h \in H$ indicate whether or not the $h$th relaxed clique is non-empty in the solution. The sets of variables $(\mathbf{x}^h, \mathbf{y}^h), h \in H$ model the $h$th relaxed clique $S_h$ in the sense that $x_i^h$ and $y_e^h$ expresses that vertex $i$ and edge $e$ belong to $G[S_h]$, respectively. The generic formulation reads as follows:

$$\min \quad \sum_{h \in H} z^h \tag{3a}$$

$$\text{s.t.} \quad \sum_{h \in H} x_i^h = 1 \quad (\text{or} \geq 1) \qquad i \in V \tag{3b}$$

$$z^h \geq x_i^h \qquad i \in V, h \in H \tag{3c}$$

$$(\mathbf{x}^h, \mathbf{y}^h) \in \mathscr{F}(G) \qquad h \in H \tag{3d}$$

$$z^h \in \{0, 1\} \qquad h \in H \tag{3e}$$

The objective (3a) minimizes the number of relaxed cliques in the solution. (3b) are the partitioning/covering constraints. Constraints (3c) ensure that $S_h = \{i \in V : x_i^h = 1\}$ is the empty set whenever $z^h = 0$. The feasibility of $S_h$ is ensured by (3d).

## 5. Branch-and-Price

Given the proposed compact formulation (3), a natural Dantzig-Wolfe decomposition can be derived as follows: The partitioning/covering constraints (3b) become the coupling constraints and the constraints (3c)–(3e) form the subproblems, identical for each block $h \in H$, and thus blocks can be aggregated (cf. Lübbecke and Desrosiers, 2005). Since each block contains the element $(\mathbf{x}^h, \mathbf{y}^h, z^h) = (\mathbf{0}, \mathbf{0}, 0)$ with cost zero and the number of blocks was chosen sufficiently large, there is no generalized convexity constraint in this Dantzig-Wolfe reformulation. Let $\Omega$ be the set of all feasible relaxed cliques. Then, the *integer master program* (IMP) is:

$$\min \sum_{S \in \Omega} \lambda_S \tag{4a}$$

$$\text{s.t.} \sum_{S \in \Omega : i \in S} \lambda_S = 1 \quad (\text{or} \geq 1) \qquad \forall i \in V \tag{4b}$$

$$\lambda_S \geq 0 \quad \text{integer} \qquad \forall S \in \Omega. \tag{4c}$$

The objective (4a) minimizes that number of relaxed cliques, (4b) are the covering/partitiong constraints, and (4c) define the domain of the variables.

In the following, the linear relaxation of (4) is referred to as the *master program* (MP). Due to the generally very large number of variables, it has to be solved with column-generation techniques (cf. Desaulniers *et al.*, 2005). The column-generation process starts from a *restricted master program* (RMP) that comprises a (typically small) subset of the variables. It is a restriction of MP and is solved, e.g., with the simplex algorithm. Let $\pi_i, i \in V$ be the dual variables associated with constraints (4b). The pricing subproblem is model (1) with weights $w_i := \pi_i$ for all $i \in V$. When one or several negative reduced-cost columns are found, i.e., a set $S \in \Omega$ with $1 - \sum_{i \in S} \pi_i < 0$, the corresponding variables are added to the RMP, the RMP is re-optimized, and the process is iterated. The MP is solved to optimality when no more negative reduced-cost columns exist. In order to solve IMP, column generation is embedded into branch-and-bound.

There are two important aspects to be described: First, the solution of the pricing problem requires problem-specific solution algorithms depending on the clique relaxation at hand. Algorithms are surveyed in the next subsection. Second, there exist competing branching schemes with the general tradeoff between either being subproblem-structure preserving and being effective in dividing the search space. We derive and discuss them in Section 5.2.

### *5.1. Pricing Algorithms*

The first type of approach is based on solving the respective MIP formulation (as the ones presented in Section 3) of the pricing subproblem (1) directly using a MIP solver. The connectivity requirements complicate the MIP-based solution, since constraints (2) have to be added in a cutting-plane (CP) fashion leading to a branch-and-cut algorithm. Table 3 therefore distinguishes between MIP and MIP-CP when giving the overview of applicable pricing algorithms for the 16 variants to be solved with model (4).

The second type of approach are combinatorial branch-and-bound (CB&B) algorithms. The *Russian doll search* (RDS) originally developed for the identification of maximal hereditary structures (Verfaillie *et al.*, 1996) is such an approach. The well-known `cliquer` algorithm for identifying maximum-weight cliques by Östergård (2002) follows the RDS principle without explicitly making the connection to RDS. Another CB&B algorithm was presented in (Held *et al.*, 2012), where the authors use `cliquer` for relatively sparse graphs and their own new algorithm for denser graphs in order to benefit from the better performance of the respective algorithm. Trukhanov *et al.* (2013) present RDS algorithms for $s$-plex and $s$-defective clique. Gschwind *et al.* (2015) revisit these RDS algorithms, suggest several techniques to accelerate the search, and

present a first exact algorithm for $s$-bundle. Overall, these algorithms cover the hereditary cases of $s$-clique, $s$-plex, $s$-defective clique, and $s$-bundle.

The RDS principle was not designed to handle connectivity. Indeed, RDS may return a solution that is disconnected, as shown by the examples given in Figure 1. Also, covering and partitioning with connected $s$-cliques for $s \geq 2$ renders the indirect solution as a clique partitioning problem in the $s$th power graph infeasible. In Section C of the Appendix, we therefore present a new adaptation of RDS, in the following referred to as *modified RDS* (mRDS), to handle connected versions of $s$-clique, $s$-plex, $s$-defective clique, and $s$-bundle. Note further that mRDS must have the ability to handle arbitrary weights $w_i \in \mathbb{R}$ for $i \in V$, since negative weights can arise in the partitioning case.

Two CB&B have been proposed for maximum cardinality $s$-club (Bourjolly *et al.*, 2002; Mahdavi Pajouh and Balasundaram, 2012). There is no need to modify the algorithms for ensuring connectivity, since an $s$-club is always connected. However, as $s$-club is not hereditary, any negative vertex weights require their careful handling in bounding procedures. A new CB&B for maximum-weight $s$-club is proposed for this purpose in Section D of the Appendix.

### *5.2. Branching*

The design of a branching scheme is crucial for the performance of a branch-and-price algorithm (Vanderbeck, 2011). First and foremost, it must ensure that integrality can be imposed in all cases. The perfect branching-scheme would be one that lets the algorithm find and prove an optimal solution quickly, i.e., it creates a small search tree, allows a fast solution of each node, and does not require modifications neither on the master problem nor the subproblem algorithm. The latter means that branching should not alter the structure of the respective pricing problem (structure preserving) so that the best performing algorithm can be applied during the entire search. Such a perfect branching scheme does not exist for partitioning and covering a graph into relaxed cliques as we explain next. We present competing branching schemes that are generic and applicable to all 16 variants allowing us to reuse all pricing algorithms presented in the previous section. We first discuss two alternative branching rules for partitioning before we reuse some of the results to introduce branching schemes for covering.

Let $\boldsymbol{\lambda}$ be a solution of the RMP. The *support graph* of $\boldsymbol{\lambda}$ is the weighted undirected graph $G^\lambda = (V, E^\lambda)$ defined by $E^\lambda := \{\{i,j\} : i,j \in V, i \neq j, f_{ij}^\lambda > 0\}$ with $f_{ij}^\lambda := \sum_{S \in \Omega: i,j \in S} \lambda_S$. It follows $f_{ij}^\lambda \geq 0$ and, for partitioning, also $f_{ij}^\lambda \leq 1$.

#### *5.2.1. Ryan-Foster Branching for Relaxed Clique Partitioning.*

Ryan-Foster branching has been proven as one of the most effective branching rules when solving set-partitioning problems with LP-based branch-and-bound (Ryan and Foster, 1981): When $\boldsymbol{\lambda}$ is a fractional solution to the LP-relaxation $A\boldsymbol{\lambda} = \mathbf{1}, \boldsymbol{\lambda} \geq \mathbf{0}$ (with binary matrix $A = (a_{ik})$), then there exist at least two rows of $A$, say $i$ and $j$, such that $0 < \sum_{k: a_{ik} = a_{jk} = 1} \lambda_k < 1$. In any integer solution, however, this value is 0 or 1. For any $i$-$j$-pair with fractional value, two branches can be created, the so-called *together-branch* forcing variables $\lambda_k$ with $a_{ik} + a_{jk} = 1$ to zero, and the *separate-branch* forcing variables $\lambda_k$ with $a_{ik} = a_{jk} = 1$ to zero. The fractional values $f_{ij}^\lambda$ allow the direct detection of branching opportunities, where the together-branch results in $f_{ij}^\lambda = 1$ and the separate-branch in $f_{ij}^\lambda = 0$.

The impact on the compact formulation (3) is additional constraints $x_i^h = x_j^h$ in the together-branch and constraints $x_i^h + x_j^h \leq 1$ in the separate-branch for all $h \in H$. For the extensive formulation (4), the subproblems must then respect $x_i = x_j$ and $x_i + x_j \leq 1$, respectively. Such constraints are simple to enforce if the subproblem is solved with a MIP solver.

Moreover, instead of considering every possible $i$-$j$-pair, Ryan-Foster branching can be implemented by using only edges $\{i,j\} \in E$ in case of connected relaxed cliques. Figure 4 shows that the connectivity requirement is crucial. The validity of this statement is straightforward to prove.

Ryan-Foster branching is a generic branching rule, since it is applicable to the MIP formulations of the subproblem. In addition, Section 5.3 shows how to apply it in situations where it is impossible or inconvenient to directly modify pricing algorithms such as CB&B. However, Ryan-Foster branching is not structure preserving.

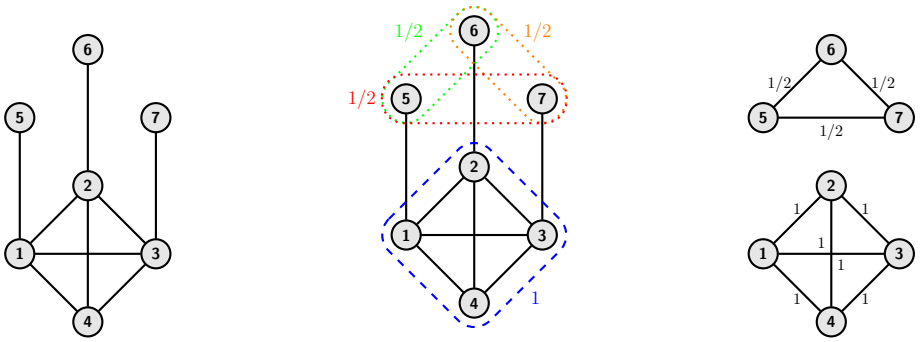


Figure 4: Partitioning with (possibly disconnected) 2-plexes. Example shows (a) the graph $G$, (b) the fractional solution $\lambda_{\{1,2,3,4\}} = 1$ and $\lambda_{\{5,6\}} = \lambda_{\{5,7\}} = \lambda_{\{6,7\}} = 0.5$, (c) the support graph having binary $f_{ij}^{\lambda}$ for all $\{i,j\} \in E$

*5.2.2. Generic Branching Rule for Relaxed Clique Partitioning.*

The here proposed *generic branching rule* (GBR) is a complete and subproblem-structure preserving rule applicable when partitioning with several relaxed clique variants. The rule is straightforward to implement because it suffices to remove edges from the given graph $G$. It does neither impose additional constraints nor require a repeated solution of the subproblem, and can be shown to be a complete branching rule for hereditary $\Pi$ in the presence of connectivity constraints.

Let $S_1, S_2, \ldots, S_p$ be the vertices inducing the connected components of $G^\lambda$, that is, $G^\lambda = G^\lambda[S_1] \cup G^\lambda[S_2] \cup \cdots \cup G^\lambda[S_p]$. When all vertex-induced subgraphs $G[S_q]$ for $q = 1, \ldots, p$ fulfill $\Pi$ no branching is required, since a feasible partition into $p$ relaxed cliques has been found. Moreover, when connectivity is required the master problem has objective $\sum_{S \in \Omega} \lambda_S = p$. Figure 4 shows that, for solutions with disconnected structures, $\boldsymbol{\lambda}$ can be fractional although all component $G[S_q]$ are feasible. In this case, $\sum_{S \in \Omega} \lambda_S < p$, and the GBR fails.

We now assume that at least one of the sets $S = S_q$ does not induce a feasible relaxed clique. We exclude the occurrence of the component $G^\lambda[S]$ via branching by removing its edges one by one. This creates $|E(G^\lambda[S])|$ branches, where in each branch just one edge is removed from $G$. The following property is helpful for drastically reducing the number of branches.

**Property 1.** *Let $\Pi$ be hereditary and connectivity be required. Given a subset $S \subseteq V$ with $G[S]$ does not fulfill $\Pi$, let $T = (S, E_T)$ be an arbitrary tree spanning $S$. Then, any feasible solution $(\mathbf{x}^h, \mathbf{y}^h, z^h)_{h \in H}$ to (3) fulfills $\sum_{h \in H} \sum_{e \in E_T} y_e^h \leq |S| - 2$.*

Proof: Since the tree contains exactly $|S| - 1$ edges, the constraint $\sum_{e \in E_T} y_e \leq |S| - 2$ means that at least one of the tree edges is not present in the solution. Otherwise, the simultaneous presence of all edges $e \in E_T$ would imply that there exists a relaxed clique $S'$ in the solution with $S' \supseteq S$. Due to the heredity of $\Pi$ this is impossible. □

If $\Pi$ is hereditary and connectivity is required, this gives rise to the GBR formalized in Algorithm 1, which guarantees that after a finite number of branchings the solution of the RMP is integral.

**Proposition 1.** *The GBR is a complete rule for partitioning into relaxed cliques with hereditary $\Pi$ and connectivity constraints.*

Proof: The requirement of $G^\lambda[S]$ to be connected ensures that a spanning tree exists. Thus, by construction, the value $f_{ij}^{\lambda}$ is strictly positive for the tree edges, cutting off the current solution $\lambda$ in each of the resulting branches. Branching can be repeated only up to $|E|$ times because each branch eliminates one edge from $G$, finally leading to an edgeless graph. Consequently, solutions $\boldsymbol{\lambda}$ must finally become integral. □

GBR is a non-binary branching scheme because generally the tree $T$ contains more than two edges. The optional Step 4 reduces the number of branches that are created. The reduction raises a new type of

**Algorithm 1:** Generic Branching Rule (GBR)

**Input** : Support graph $G^\lambda$ and weights $f^\lambda_{ij}$

1 Determine the connected components $S_1, S_2, \ldots, S_p$ of $G^\lambda$.
2 Identify one component $S$ for which $G[S]$ does not fulfill $\Pi$.
3 If no such component exists, then stop (the solution is already a union of feasible relaxed cliques).
4 (Optional) Replace $S$ by one of its subsets such that
(a) $G[S]$ does not fulfill $\Pi$, (b) $G^\lambda[S]$ is connected, and (c) $|S|$ is minimal.
5 Determine a maximum weight spanning tree $T = (S, E(T))$ of $G^\lambda[S]$ using the weights $f^\lambda_{ij}$.

**Output** : $E(T)$, the set of edges to eliminate one by one

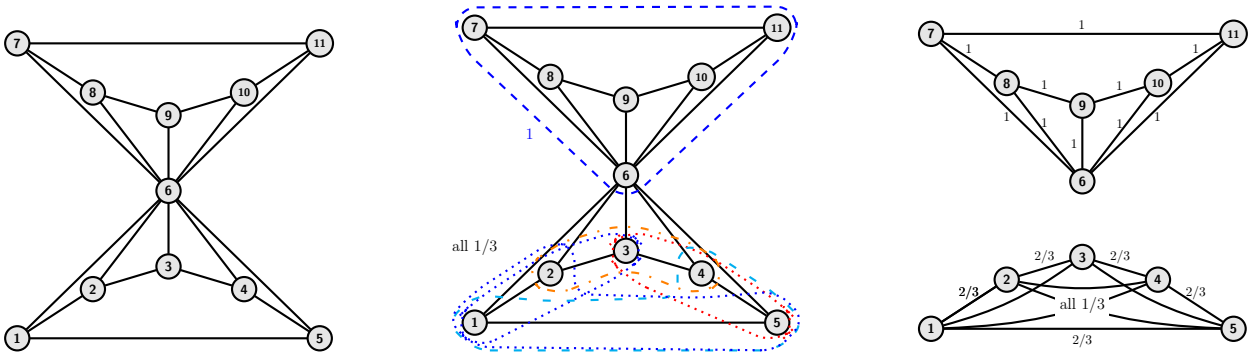


Figure 5: Partitioning with 2/3-quasi-cliques. (a) the graph $G$, (b) the factional solution $\lambda_{\{6,7,8,9,10,11\}} = 1$ and $\lambda_{\{1,2,3\}} = \lambda_{\{2,3,4\}} = \lambda_{\{3,4,5\}} = \lambda_{\{1,4,5\}} = \lambda_{\{1,2,5\}} = 1/3$, (c) the support graph in which the component $S = \{1, 2, 3, 4, 5\}$ is no 2/3-quasi-clique, but $S' = S \cup \{6\} = \{1, 2, 3, 4, 5, 6\}$ is a 2/3-quasi-clique

optimization problem that we briefly discuss in Section E of the Appendix. The computation of the spanning tree in Step 5 is computationally cheap (using Prim's or Kruskal's algorithm). The selection of maximum weight edges is intended to produce branches that improve the lower bounds as much as possible.

*The non-hereditary case..* A prerequisite of GBR as presented in Algorithm 1 is that $\Pi$ is hereditary. However, even for non-hereditary $\Pi$ a modified version of GBR can be used. Note that heredity of $\Pi$ has only been exploited in order to ensure that no superset of $S' \supsetneq S$ is a feasible relaxed clique. If $S$ has no such superset and the optional reduction in Step 4 of GBR is skipped, branching on the edges of the tree spanning $G^\lambda[S]$ is valid.

However, two drawbacks have to be pointed out: First, the average number of branches created with GBR can be expected to be larger due to the skipped reduction step. Second, and more serious is that, for non-hereditary $\Pi$, GBR may not be applicable at all and, thus, it is incomplete. This happens if all connected components which do not fulfill $\Pi$ have a superset satisfying $\Pi$. Figure 5 provides an example when partitioning with 2/3-quasi-cliques.

For $s$-club, we can derive a simple branching rule creating exactly $s + 1$ branches in each step: If a component $S$ is infeasible, there must exist two vertices $i, j \in V$ with $\text{dist}_{G[S]}(i, j) = s + 1$. If these vertices also fulfill $\text{dist}_G(i, j) = s + 1$ (recall that generally $\text{dist}_{G[S]}(i, j) \geq \text{dist}_G(i, j)$ holds), then $S$ can be chosen as $V(P_{ij})$, where $P_{ij}$ is a path of length $s + 1$ connecting $i$ with $j$ in $G$. It remains an open question whether this variant of the GBR is complete. During our experiments, we never found an example (as the one for quasi-clique shown in Figure 5) for $s$-club and any $s \geq 2$.

The column *Partitioning* of Table 4 summarizes the possible branching schemes presented in this section.

*5.2.3. Generic Branching Rules for Relaxed Clique Covering.*

For covering with relaxed cliques, we propose a multi-level branching scheme, where the first two levels decide on so-called vertex contacts and the lower levels assure integrality and apply branching rules known for partitioning, i.e., Ryan-Foster branching or the GBR.

Table 4: Branching schemes

| | Partitioning | | | Covering |
|---|---|---|---|---|
| Branching scheme | `P1`:<br>\| `1.` Ryan Foster | `P2`:<br>\| `1.` GBR | `P3`:<br>\| `1.` GBR<br>\| `2.` Ryan Foster | `C1`–`C3`:<br>\| `1.` Vertex contacts fractional<br>\| `2.` Vertex contacts fixation<br>\| `3.` `P1` or `P2` or `P3` on $V_{=1}$<br>\| `4.` Vertex duplication |
| Applicable to | $s$-plex<br>$s$-clique<br>$s$-club<br>$\gamma$-quasi-clique<br>$s$-defective clique<br>$s$-bundle | $s$-plex, connected<br>$s$-clique, connected<br>$s$-defective, connected<br>$s$-bundle, connected | $s$-plex, general<br>$s$-clique, general<br>$s$-club$^{\ddagger}$<br>$\gamma$-quasi-clique<br>$s$-defective, general<br>$s$-bundle, general | *all variants* |

Branching schemes `C1`, `C2`, and `C3` for covering results from the use of appropriate branching rules `P1`, `P2`, and `P3`, respectively, at level `3.` of the scheme. ‡: If GBR is a complete branching rule for $s$-club, `P2` is applicable instead of `P3`.

For any subset $T \subseteq V$, the number of *vertex contacts* is defined as $g(T) = \sum_{S \in \Omega} |T \cap S| \lambda_S$. In order to distinguish between variables and values, we write $g(T)$ for the sum of the variables and $g^\lambda(T)$ for the resulting value. For the sake of convenience, we also define $g_i = g(\{i\})$ and $g_i^\lambda = g^\lambda(\{i\})$ for vertices $i \in V$. Branching on the number of vertex contacts preserves the structure of the subproblem. Indeed, enforcing $g(T) \leq \lfloor g^\lambda(T) \rfloor$ or $g(T) \geq \lceil g^\lambda(T) \rceil$ only changes the weights $w_i, i \in T$ in the subproblem's objective according to the dual price of the constraint. In case of less-or-equal inequalities, weights can become negative.

It may happen that all vertices $i \in V$ have integer vertex contacts $g_i^\lambda$, but the solution $\lambda$ is still fractional. Such a situation is certainly not unusual because in the set-partitioning case all vertex contacts are equal to one and fractional solutions are predominant. Therefore, additional branching actions need to be taken (in the following referred to as *vertex contact fixation*). The intention of our higher-level branching is to fix, for a large subset $P \subseteq V$, the vertex contacts to its minimum, i.e., $g(P) = |P|$, in order to then treat this subset as in the partitioning case. Beside fixation, an alternative branch $g(P) \geq |P| + 1$ must be created, too. Deeper in the tree, branches $g(P) = |P| + 1$ and $g(P) \geq |P| + 2$, and generally $g(P) = |P| + p$ and $g(P) \geq |P| + p + 1$ are created. Note that $g(P) \geq |V| + 1$ is certainly suboptimal so that this process always stops.

If $g(P)$ is fixed as well as all vertex contacts of vertices in $P$, i.e., all $g_i$ for all $i \in P$ are fixed to specific values, then one can proceed as follows. All vertices with their vertex contacts fixed to 1 form the set $V_{=1} = \{i \in V : g_i \text{ is fixed to } 1\}$. Ryan-Foster branching or the GBR can be applied to $V_{=1}$ for which the smaller support graph $G^\lambda[V_{=1}]$ must be considered. If no branching is possible, a final graph modification procedure can always be applied (referred to as *vertex duplication*). The remaining vertices $V_{>1} = \{i \in V : g_i \text{ is fixed to a value} > 1\}$ are replaced by exactly $g_i$ clones $i^1, \ldots, i^{g_i}$. The clones are adjacent to exactly the same vertices as $i$. Moreover, the additional separate-constraints that no two clones $i^k$ and $i^\ell$ for $k \neq \ell$ occur together in a relaxed clique are imposed. In all experiments, it was never necessary to apply vertex duplication, since solutions were already integral.

The column *Covering* of Table 4 summarizes the branching schemes for the covering variants.

### *5.3. Generic Handling of Together- and Separate-Constraints in Pricing*

For vertex coloring (clique partitioning of the complement graph), the Ryan-Foster constraints can be imposed solely by graph modifications. In the together-branch, the vertices $i$ and $j$ are merged, while in the separate-branch the new edge $\{i, j\}$ is added (Mehrotra and Trick, 1998; Held *et al.*, 2012). However, for relaxed cliques, such graph modifications generally change distance, density, and connectivity on many subsets $S$ and, therefore, do not reproduce the original situation with the separate/together constraint added.

A single separate-constraint can be implemented by solving two different subproblems, since the removal of one vertex, either $i$ or $j$, ensures that the two vertices never occur together in a relaxed clique. A removal of vertex $i$ is equivalent to assigning a huge negative weight $w_i = -M$ (using a big-$M$). Also a

Table 5: Instance features

| $G=(V,E)$ | \|V\| | \|E\| | $\rho(G)$ | $\delta(G)$ | $\alpha(G)$ | $\omega(G)$ | $\chi(G)$ | $\chi(\bar{G})$ |
|---|---|---|---|---|---|---|---|---|
| karate | 34 | 78 | 0.1390 | 1 | 20 | 5 | 5 | 20 |
| chesapeake | 39 | 170 | 0.2294 | 3 | 17 | 5 | 5 | 17 |
| dolphins | 62 | 159 | 0.0841 | 1 | 28 | 5 | 5 | 28 |
| lesmis | 77 | 254 | 0.0868 | 1 | 35 | 10 | 10 | 35 |
| polbooks | 105 | 441 | 0.0808 | 2 | 43 | 6 | 6 | 43 |
| adjnoun | 112 | 425 | 0.0684 | 1 | 53 | 5 | 5 | 55 |
| football | 115 | 613 | 0.0935 | 7 | 21 | 9 | 9 | 22 |
| jazz | 198 | 2742 | 0.1406 | 1 | 40 | 30 | 30 | 40 |
| celegansneural | 297 | 2148 | 0.0489 | 1 | 110 | 8 | [8,9] | 115 |

single together-constraint for $i$ and $j$ can be realized by solving two different subproblems, where in one subproblem both vertices are removed, while in the other both are enforced by assigning a huge positive weight $w_i = w_j = M$.

Although this approach seems appealing, it has the drawback that when $q$ Ryan-Foster constraints are active up to $2^q$ different subproblems have to be solved. In order to mitigate this explosion, we propose the following branch-and-bound algorithm. At its root node, no constraints are active. At each node, the relaxed subproblem is solved and if infeasible, a single separate- or together-constraint creates two new branches. A depth-first node selection is applied in order to have lower bounds available at an early stage. Note that this branch-and-bound approach is truly generic as it can be used in combination with any of the subproblem algorithms (see Section 5.1 and last column of Table 3).

## 6. Computational Results

The results reported in this section were obtained using a single thread of a standard PC with an Intel(R) Core(TM) i7-4790 3.6 GHz processor and 8 GB of main memory. The algorithms were coded in C++ and compiled with MS Visual Studio 2010. The callable library of CPLEX 12.5 was used for solving all LPs and MIPs.

Since the number of considered problems is already large (16 variants of partitioning and covering with values $s \in \{2,3,4,5\}$ and $\gamma = \{.95,.9,.85,.8,.75\}$), we have restricted our computational analysis to the nine networks from the 10th DIMACS challenge with less than 300 vertices (available at `http://dimacs.rutgers.edu/Challenges/`). This gives rise to an overall of 1296 instances. Table 5 lists the nine networks $G=(V,E)$ and their characteristics: density $\rho(G)$, minimum degree $\delta(G)$, maximum independent set size $\alpha(G)$, maximum clique size $\omega(G)$, chromatic number $\chi(G)$, and chromatic number of the complement graph $\chi(\bar{G})$. Note that unlike for maximum cardinality relaxed cliques, no graph reduction by a peeling procedure is possible when decomposing the entire network. For all experiments, the computation time is limited to 600 seconds.

Recall that for hereditary $\Pi$ (non-connected), partitioning and covering are equivalent. Hence, we solve a set-covering master program in which the dual values are more stable and post-process the solution to obtain a feasible partitioning. In order to stabilize the column-generation process also in the proper partitioning cases, we replace partitioning by covering constraints (4b) and add the additional constraint that the number of vertex contacts must not exceed $n = |V|$, i.e., $\sum_{S\in\Omega} |S|\lambda_S \leq n$, to the master program (4). The effect is that all dual prices $\pi_i$ for $i \in V$ are non-negative. The resulting weight for vertex $i \in V$ is then $w_i := \pi_i + \mu$, where $\mu$ is the (non-positive) dual price of the added constraint.

Moreover, we use a multi-column pricing strategy: For MIPs, all integer feasible solutions with negative reduced cost found by CPLEX are added to the master. Similarly, all different solutions found in the main loop of (m)RDS are added. For $s$-club, we use a different acceleration strategy, i.e., the CB&B prematurely stops as soon as a solution with reduced cost smaller than $-0.1$ is found.

Table 6: Comparison of pricing algorithms

| | | | $s=2$ | $s=3$ | $s=4$ | $s=5$ | |
|---|---|---|---|---|---|---|---|
| $s$-plex | Partitioning | Connected | 405.5 | 150.1 | 35.4 | 4.3 | |
| | Covering | Connected | 319.5 | 103.6 | 26.0 | 6.3 | |
| | Part./Cover. | General | 269.7 | 36.4 | 4.0 | 0.6 | |
| $s$-club | Partitioning | General | 57.3 | 124.1 | 153.7 | 215.6 | |
| | Covering | General | 45.9 | 79.4 | 116.8 | 326.8 | |
| $s$-bundle | Partitioning | Connected | 525425.1 | 89556.1 | 18944.2 | 2772.4 | |
| | Covering | Connected | 1374606.2 | 154631.5 | 15714.0 | 3112.4 | |
| | Part./Cover. | General | 390207.8 | 124293.4 | 14491.8 | 2169.8 | |
| | | | $\gamma=0.95$ | $\gamma=0.90$ | $\gamma=0.85$ | $\gamma=0.80$ | $\gamma=0.75$ |
| $\gamma$-quasi-clique | Partitioning | Connected | 5.7 | 2.7 | 1.3 | 0.9 | 0.8 |
| | Covering | Connected | 6.5 | 3.0 | 1.4 | 0.9 | 0.7 |
| | Partitioning | General | 4.0 | 2.3 | 1.0 | 1.0 | 0.7 |
| | Covering | General | 4.5 | 2.5 | 1.1 | 0.9 | 0.7 |

Factors are the average ratios $t_{PP}^{MIP}/t_{PP}^{CB\&B}$ and $t_{PP}^{F1}/t_{PP}^{F2}$

*6.1. Linear Relaxation Results*

In a first series of experiments, we analyze the performance of alternative pricing algorithms (see Section 5.1). For $s$-plex, we compare the IP formulation of Balasundaram *et al.* (2011) with the (m)RDS (see Section C of the Appendix). For $s$-club, we compare the IP formulation of Veremyev and Boginski (2012) with our CB&B presented in Section D of the Appendix. For $s$-bundle, we compare our MIP formulation presented in Section B of the Appendix with the (m)RDS. Finally, for $\gamma$-quasi-clique, we compare the two MIP formulations of Pattillo *et al.* (2013b). When connectivity is required, we impose it in all MIPs by separating constraints (2).

Table 6 presents aggregated results over all nine benchmark networks. The numbers are ratios of the average times that a single pricing iteration consumes. Whenever the linear relaxation of (4) is not completely solved within the time limit, the average is taken over the iterations solved up to this point. For $s$-bundle, CPLEX is only able to solve the pricing problem for the two smallest instances `karate` and `chesapeake` (the others caused an out-of-memory exception). When comparing MIP with CB&B, a ratio $t_{PP}^{MIP}/t_{PP}^{CB\&B}$ of more than 1 indicates that CPLEX needs more time than the respective CB&B, while for $\gamma$-quasi-clique a ratio $t_{PP}^{F1}/t_{PP}^{F2}$ greater than 1 means that CPLEX takes longer for solving the first MIP (`F1`) than for the second MIP (`F2`) of Pattillo *et al.* (2013b).

Overall, the CB&B algorithms perform better than the MIPs and with increasing $s$ the effect becomes less pronounced for $s$-plex and $s$-bundle. In contrast, the results for $s$-club show that the MIP-based approach becomes less attractive when $s$ increases. For $\gamma$-quasi-clique, none of the two MIPs completely dominates the other. Therefore, in all following experiments we solve `F1` whenever $\gamma \leq 0.8$ and `F2` in all other cases. The results also seem to indicate that for general $s$-plex and $s \geq 5$ pricing with MIP is superior to RDS. For consistency among different $s$ values, however, all following results are computed with RDS.

In Table 7, we present absolute computation times for solving the linear relaxations. Numbers is brackets show the number of instances for which the linear relaxation is solved within the time limit; (*) means that all nine instances are solved. If an instance is not solved, it contributes to the presented average with 600 seconds. In all cases, pricing consumes more than 99% of the computation time.

The column-generation algorithm for covering with $s$-club is able to solve all 36 linear relaxations. For the other problem variants, the algorithms are not able to solve all instances for all values of $s$ or $\gamma$. The hardest variants are those for $\gamma$-quasi-clique, while for the hereditary structures almost all instances are solved with $s = 2$ and $s = 3$. Larger values of $s$ and smaller values of $\gamma$ lead to harder to solve pricing problems, larger generated relaxed cliques, more degenerate master programs typically requiring more iterations, and herewith to longer computation times for the linear relaxation. An exception are the

Table 7: Linear programming relaxation average computation times

| | | | $s=2$ | $s=3$ | $s=4$ | $s=5$ | |
|---|---|---|---|---|---|---|---|
| $s$-plex | Partitioning | Connected | 0.7 (*) | 47.8 (*) | 168.3 (7) | 360.4 (4) | |
| | Covering | Connected | 0.5 (*) | 39.9 (*) | 161.6 (7) | 351.3 (4) | |
| | Part./Cover. | General | 0.5 (*) | 38.9 (*) | 168.2 (7) | 350.9 (4) | |
| $s$-clique | Part./Cover. | Connected | 67.4 (8) | 66.9 (8) | 0.1 (*) | 0.1 (*) | |
| $s$-club | Partitioning | General | 214.1 (7) | 372.8 (5) | 335.7 (4) | 95.9 (8) | |
| | Covering | General | 3.5 (*) | 8.7 (*) | 0.1 (*) | 0.1 (*) | |
| $s$-defective | Partitioning | Connected | 0.5 (*) | 3.1 (*) | 41.3 (*) | 86.2 (8) | |
| clique | Covering | Connected | 0.4 (*) | 2.2 (*) | 39.4 (*) | 76.2 (8) | |
| | Part./Cover. | General | 0.4 (*) | 2.4 (*) | 32.0 (*) | 78.9 (8) | |
| $s$-bundle | Partitioning | Connected | 0.8 (*) | 53.7 (*) | 175.3 (7) | 362.7 (4) | |
| | Covering | Connected | 0.6 (*) | 43.4 (*) | 172.4 (7) | 347.5 (4) | |
| | Part./Cover. | General | 0.6 (*) | 40.4 (*) | 166.2 (7) | 355.9 (4) | |
| | | | $\gamma=0.95$ | $\gamma=0.90$ | $\gamma=0.85$ | $\gamma=0.80$ | $\gamma=0.75$ |
| $\gamma$-quasi- | Partitioning | Connected | 279.7 (6) | 356.6 (4) | 395.4 (4) | 434.5 (3) | 444.1 (3) |
| clique | Covering | Connected | 251.7 (6) | 349.5 (4) | 366.7 (4) | 434.7 (3) | 435.6 (3) |
| | Partitioning | General | 314.1 (6) | 357.2 (4) | 400.3 (4) | 427.5 (3) | 434.5 (3) |
| | Covering | General | 261.5 (6) | 345.2 (4) | 369.9 (4) | 405.2 (4) | 426.9 (3) |

distance-based relaxations $s$-clique and $s$-club, where for larger $s$ the decomposition becomes trivial because the given graph is already an $s$-clique/club. Among the other hereditary structures, the linear relaxation for $s$-defective clique is solved faster than for $s$-plex and $s$-bundle, which seem to be similar. The latter result is somewhat unexpected when comparing with the results of Gschwind *et al.* (2015) where maximum cardinality $s$-bundle was harder than $s$-plex.

The only variant for which partitioning is much more time consuming than covering is $s$-club: the presence of some negative weights seems to substantially complicate the pricing. We observe that single instances of the pricing problem require significantly more time than the average. The results for the other variants show that covering is slightly easier than partitioning, but the differences are not substantial. We also observe that pricing consumes more time for partitioning due to some negative weights, but the multiple-pricing strategy at the same time produces more relaxed cliques leading to a comparable number of pricing iterations. Comparing covering with connected relaxed cliques and decomposing with general relaxed cliques (both formulated as a set covering master) shows that computation times are strongly correlated.

### *6.2. Integer Results*

For $s$-plex, $s$-club, and $s$-bundle, we tried to decompose the networks using the compact formulation (3). Results were very disappointing, since not even the smallest network `karate` could be decomposed even after providing the optimal number $rc(G)$ of necessary relaxed cliques. The poor performance can be attributed to the weak lower bound provided by the linear relaxation and the inherent symmetry of the compact formulation.

We next analyze the performance of the branching rules of Section 5.2 for partitioning into relaxed cliques. Depending on the problem variant, we compare `P1` (Ryan Foster) against `P2` (GBR) or `P3` (GBR followed by Ryan Foster). The node-selection strategy is depth-first in order to find upper bounds early in the search.

Table 8 shows average computation times for solving the integer model. As before, numbers in brackets indicate the number of instances solved to proven optimality with (*) when all instances are solved. Furthermore, for those instances solved to optimality with both branching schemes, Table 9 gives the size of the branch-and-bound tree (minimum, average, and maximum over the instances).

The branching rules based on GBR are clearly inferior: Average computation times of `P1` are always smaller than those of `P2` and `P3`. There are, however, a few instances for which `P1` takes longer. Scheme

Table 8: Comparison of branching schemes for relaxed clique partitioning: Average computation time and number of optimal solutions

| | | $s=2$ | | $s=3$ | | $s=4$ | | $s=5$ | |
|---|---|---|---|---|---|---|---|---|---|
| | | P1 | P2/P3 | P1 | P2/P3 | P1 | P2/P3 | P1 | P2/P3 |
| $s$-plex | Connected | 133.8 (7) | 210.4 (6) | 230.1 (6) | 357.5 (4) | 344.7 (4) | 541.5 (1) | 443.2 (3) | 546.2 (2) |
| | General | 133.9 (7) | 200.2 (6) | 217.7 (6) | 297.0 (5) | 345.7 (4) | 413.8 (3) | 473.4 (2) | 475.4 (2) |
| $s$-clique | Connected | 68.0 (8) | 133.9 (7) | 66.9 (8) | 66.9 (8) | 0.1 (*) | 0.1 (*) | 0.1 (*) | 0.1 (*) |
| $s$-club | General | 335.5 (4) | 337.9 (4) | 401.3 (3) | 401.3 (3) | 335.6 (4) | 335.6 (4) | 94.6 (8) | 94.7 (8) |
| $s$-defective | Connected | 76.3 (8) | 335.0 (4) | 155.0 (7) | 268.0 (5) | 214.7 (6) | 400.8 (3) | 278.7 (5) | 351.4 (4) |
| | General | 72.7 (8) | 218.1 (6) | 169.8 (7) | 268.7 (5) | 219.4 (6) | 339.6 (4) | 241.0 (6) | 412.2 (3) |
| $s$-bundle | Connected | 133.8 (7) | 210.9 (6) | 184.0 (7) | 401.2 (3) | 347.7 (4) | 472.9 (2) | 444.7 (3) | 521.4 (2) |
| | General | 133.9 (7) | 200.3 (6) | 258.4 (6) | 336.5 (4) | 358.2 (4) | 419.0 (3) | 477.3 (2) | 498.8 (2) |

Table 9: Comparison of branching schemes for relaxed clique partitioning: Tree size (min/avg/max)

| | | $s=2$ | | $s=3$ | | $s=4$ | | $s=5$ | |
|---|---|---|---|---|---|---|---|---|---|
| | | P1 | P2/P3 | P1 | P2/P3 | P1 | P2/P3 | P1 | P2/P3 |
| $s$-plex | Connected | 5/11/31 | 3/1007/5701 | 1/12/19 | 1/972/3345 | 19/19/19 | 537/537/537 | 10/15/20 | 256/729/1201 |
| | General | 1/9/20 | 1/16/37 | 9/17/22 | 7/491/2359 | 9/17/21 | 13/468/1343 | 23/25/26 | 63/81/99 |
| $s$-clique | Connected | 1/1/4 | 1/1/4 | 1/1/1 | 1/1/1 | 1/1/1 | 1/1/1 | 1/1/1 | 1/1/1 |
| $s$-club | General | 1/2/2 | 1/34/48 | 1/1/1 | 1/1/1 | 1/1/1 | 1/1/1 | 1/1/1 | 1/1/1 |
| $s$-defective | Connected | 5/12/18 | 8/1086/4241 | 2/6/9 | 2/109/365 | 5/523/1550 | 27/121/294 | 5/14/25 | 4/758/2675 |
| | General | 11/3059/18151 | 32/2275/11734 | 7/17/24 | 11/62/217 | 4/16/21 | 20/51/79 | 11/22/39 | 38/46/51 |
| $s$-bundle | Connected | 5/11/31 | 3/1007/5701 | 1/4/6 | 1/29/83 | 15/17/18 | 214/259/304 | 20/21/22 | 291/333/375 |
| | General | 1/9/20 | 1/16/37 | 7/13/26 | 2/115/423 | 21/28/42 | 41/61/91 | 14/20/26 | 48/91/133 |

P1 is superior also with respect to the number of optima. All instances solved with P2 or P3 are also solved with P1. An explanation for this outcome is that GBR-based rules create, with a few exceptions, many more branches than the pure Ryan-Foster rule, see Table 9. Analyzing times and tree sizes together reveals that for GBR-based rules a single branch-and-bound node is solved faster. This is intuitive because GBR is a structure-preserving rule as opposed to the Ryan-Foster rule which requires the use of less effective pricing algorithms (see Section 5.3).

Based on these findings, the final series of experiments applies branching scheme P1 for partitioning and the corresponding scheme C1 for covering problems. Since linear relaxation bounds are generally tight, the overall performance of our algorithms very much depends on the ability to find good feasible decompositions (upper bounds) fast. Therefore, we solve the master program as an integer model with CPLEX at every branch-and-bound node. Pre-tests have shown that such a heuristic is particularly helpful for covering variants which often require massive branching before reaching an integer solution. In order to avoid long MIP runs, CPLEX is limited 10 seconds. Moreover, we change the node-selection rule in our branch-and-price algorithms to best-first search.

Table 10 is organized as Table 8 and displays the average computation times and the number of optima for the branch-and-price. With the help of the upper bounds provided by CPLEX, 383 instances are solved to proven optimality compared to only 356 optima without using the upper bounds. The upper bounds seem to be particularly helpful for larger values of $s$. This is also true for variants with intricate subproblems such as $s$-bundle and $\gamma$-quasi-cliques. Overall, computation times are also slightly reduced.

In summary, decomposing into relaxed cliques is a computationally challenging problem. The difference is more between different types of relaxed cliques than between partitioning and covering and between general and connected relaxed cliques. A main indicator for the hardness of the decomposition is the hardness of the corresponding pricing problem. It is therefore not surprising that the decomposition with $s$-clique and $s$-defective clique works better than with $\gamma$-quasi-clique. Finally, the determination of clusters that maximize modularity, as done by Aloise *et al.* (2010), seems at least comparably challenging as decomposition into relaxed cliques because computation times of their fastest algorithm quickly grow with the network size (they allow a maximum of 100 000 seconds). Even more, modularity maximization produces a single solution only

Table 10: Branch-and-price average computation time and number of optimal solutions found

| | | | $s = 2$ | $s = 3$ | $s = 4$ | $s = 5$ | |
|---|---|---|---|---|---|---|---|
| $s$-plex | Partitioning | Connected | 136.9 (7) | 233.2 (6) | 328.8 (5) | 395.4 (4) | |
| | Covering | Connected | 133.5 (7) | 333.6 (4) | 289.1 (5) | 348.1 (4) | |
| | Part./Cover. | General | 133.4 (7) | 139.1 (7) | 219.0 (6) | 452.5 (3) | |
| $s$-clique | Part./Cover. | Connected | 67.2 (8) | 66.8 (8) | 0.1 (*) | 0.1 (*) | |
| $s$-club | Partitioning | General | 336.4 (4) | 401.4 (3) | 335.8 (4) | 97.6 (8) | |
| | Covering | General | 85.2 (8) | 9.9 (*) | 0.1 (*) | 0.1 (*) | |
| $s$-defective | Partitioning | Connected | 145.3 (7) | 135.1 (7) | 201.0 (6) | 154.4 (7) | |
| clique | Covering | Connected | 179.4 (7) | 133.7 (7) | 137.0 (7) | 141.7 (7) | |
| | Part./Cover. | General | 136.5 (7) | 136.0 (8) | 200.8 (6) | 78.7 (8) | |
| $s$-bundle | Partitioning | Connected | 136.9 (7) | 180.7 (7) | 285.5 (5) | 435.9 (3) | |
| | Covering | Connected | 133.5 (7) | 335.0 (5) | 232.6 (7) | 403.8 (3) | |
| | Part./Cover. | General | 133.4 (7) | 247.6 (6) | 280.7 (5) | 350.3 (4) | |
| | | | $\gamma = 0.95$ | $\gamma = 0.90$ | $\gamma = 0.85$ | $\gamma = 0.80$ | $\gamma = 0.75$ |
| $\gamma$-quasi- | Partitioning | Connected | 286.2 (6) | 352.6 (4) | 385.9 (4) | 427.5 (3) | 435.1 (3) |
| clique | Covering | Connected | 243.0 (6) | 347.0 (4) | 362.6 (4) | 429.0 (3) | 430.7 (3) |
| | Partitioning | General | 300.4 (6) | 353.5 (4) | 406.8 (4) | 422.3 (3) | 427.9 (3) |
| | Covering | General | 252.3 (6) | 343.4 (4) | 363.5 (4) | 421.6 (3) | 423.3 (3) |

and seems to be less precise with respect to the true number of clusters. Unfortunately, Aloise *et al.* (2010) do not provide the actual partitions so that one can hardly check whether clusters match known features, as analyzed in the next section.

### *6.3. Interpretation of Results in Social Networks*

In this section, we test the applicability of our graph decomposition methods for the purpose of detecting community structures (cf. Fortunato, 2010). We have chosen `karate`, `dolphins`, and `football` as three prominent and intensively studied examples of social networks for which different methods of community detection have be tested.

#### *6.3.1. Zachary's Karate Club.*

Zachary (1977) introduced the formal description of a university-based karate club as an example of a fission of a small anthropological group. The relevant background information is that due to a longer-lasting conflict between the club president and the karate instructor the club finally separated into two new clubs, one supporting the old club's president and the other one following the instructor. Zachary's study, however, focused on the social interaction between members before the fission. He collected the information 'if two individuals consistently were observed to interact outside the normal activities of the club'. The crisis in the club had the effect of 'pulling apart the (sub)networks of friendship ties'. The resulting social network has one vertex for each active member of the club, and two vertices are adjacent if and only if the corresponding members consistently interacted. This network became a useful benchmark for community detection approaches, since algorithmically computed clusters can be compared with the real memberships in one of the two clubs after the division.

Moreover, Zachary measured the degree of interaction between members in the form of edge weights (using an ordinary scale between 1 and 8; we and the majority of methods from community detection do not exploit this additional data). He demonstrated that the two conflicting groups (the two clubs after the formal separation) can be identified using a max-flow min-cut computation on the edge-weighted graph (with accuracy 97%, one vertex being misclassified).

The unweighted karate club network, depicted in Figure 6(a), was used by several researchers to test their approaches. For example, Girvan and Newman (2002) applied a hierarchical clustering via tree decom-

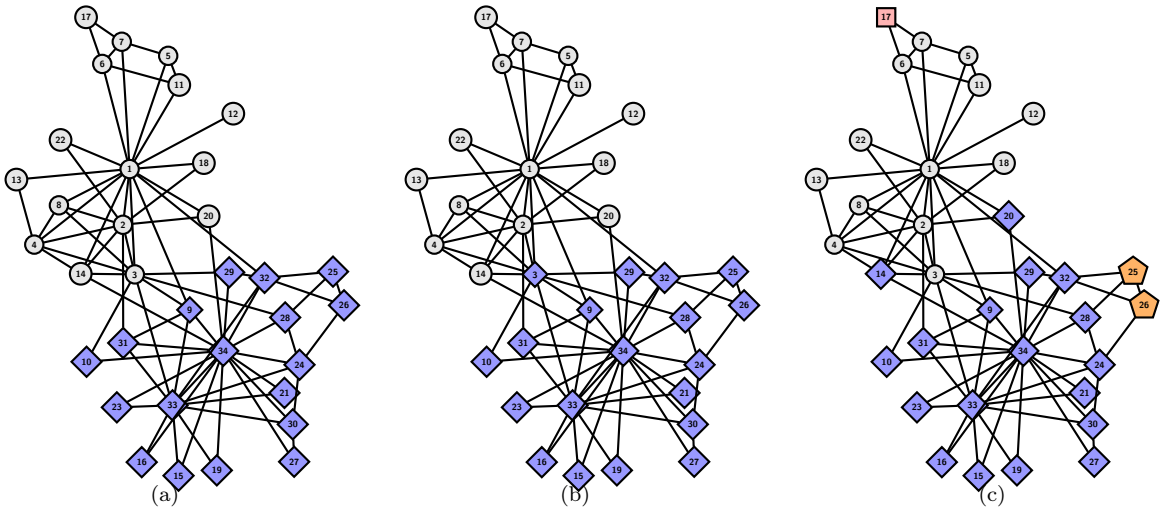

Figure 6: Zachary's karate club (a) Solution as given in (Zachary, 1977); (b) 3-club partitioning; (c) 2-club partitioning. Note that the graph is a 5-club

position. Their first split 'corresponds almost perfectly with the actual division of the club members' with only vertex 3 being misclassified (Girvan and Newman, 2002, p. 7823).

The same authors, Newman and Girvan (2004), later introduced modularity in order to measure the quality of a decomposition (see Section 1.1). Their decomposition method (i) calculates the so-called *betweenness* for all edges of the network, (ii) removes an edges with maximum betweenness, (iii) repeats the steps (i) and (ii) for the resulting reduced network until it is edgeless. This creates a hierarchical decomposition of a graph, often displayed using a decomposition tree. With the *shortest-path betweenness*, the method produces a first decomposition into two components with vertex 3 incorrectly classified, while with *random-walk betweenness* the two groups are identified correctly. However, with both decomposition methods the clustering into five/four groups achieves a higher modularity.

With the knowledge that the new clubs were formed around the polarizing persons that brought the conflict into the club, it seems natural to us to decompose the graph using a distance-based clique relaxation: The subgroups should have the property that any two members are close to a central person and, therefore, the two members must also be in close distance from each other. Moreover, the resulting subgroups should be connected. Also Almeida and Carvalho (2013) suggest the use of $s$-clubs in SNA arguing that 'social relations are frequently established through intermediaries'.

The results of a decomposition into $s$-clubs ($s = 2$ or 3) are shown in Figure 6(b) and (c). Note that vertex 1 is the club's president and vertex 34 is the instructor. The depicted solutions are at the same time solutions to the covering and partitioning problems. The mismatch of vertex 3 in the 3-club partitioning is actually by chance because the two real groups also form a decomposition into 3-clubs.

Moreover, the decomposition into four 2-clubs as depicted in Figure 6(c) is not unique, but fits with the four clusters determined using the method of Newman and Girvan (2004) with random-walk betweenness. Furthermore, the 2-club $\{17\}$ can be enlarged to $\{5, 6, 7, 11, 17\}$, which is one of the clusters identified by Newman and Girvan (2004). The same holds for the 2-club $\{25, 26\}$, which can be extended to $\{25, 26, 29, 32\}$ without making the other 2-clubs infeasible in the resulting partitioning.

*6.3.2. Dolphins.*

We next consider a network of 62 bottlenose dolphins living in Doubtful Sound (New Zealand). Lusseau (2003) defined the edges of the network as indicators of 'preferred companionships' meaning that pairs of dolphins were seen together more often than expected by chance. Figure 7(a) depicts the network. After dolphin *SN100* left the place for some time, the dolphins separated into two groups (Lusseau and Newman, 2004) indicated with the two colors.

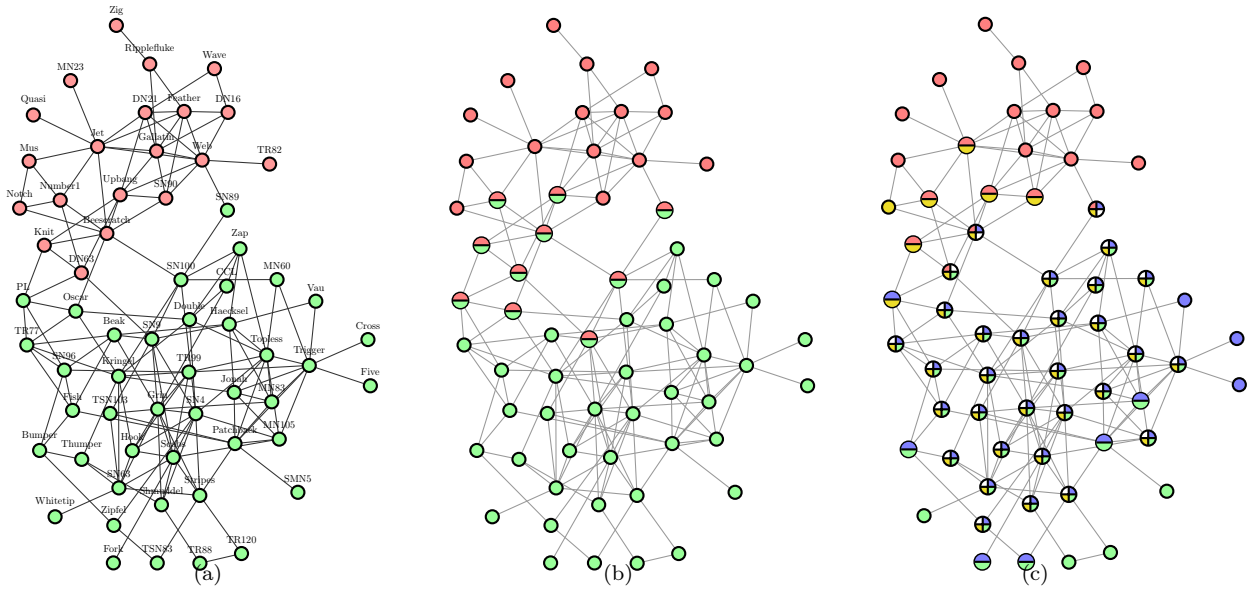


Figure 7: Dolphins (a) Real split (b) Covering with connected 5-cliques, two clusters (c) Covering connected 4-cliques, four clusters

Similar to the karate club, the dolphin network is an example of a social network in which fission and fusion was observed. As the connection between the dolphins is rather lose, a distance-based clique relaxation seems appropriate for a decomposition. Moreover, the network is larger but less dense compared to the karate club (see Table 5) so that we have chosen larger values of the maximum distance $s$. As Fortunato (2010, Section 11) points out, the identification of overlapping clusters is an important task in community detection. We now show that interesting and interpretable results can be obtained with our graph covering algorithms.

Figure 7(b) shows a decomposition of the dolphins network into connected 5-cliques that are allowed to overlap. Two communities result and are indicated by the red and green colors. The dolphins displayed with bicolored vertices are those that belong to both communities. To be precise, we computed a (non-unique) covering solution and extended each of the two communities to the depicted cardinality-maximal 5-cliques.

Obviously, the result shown in Figure 7(b) perfectly matches the real split into two communities as described in (Lusseau and Newman, 2004). Moreover, our intersection that consists of ten dolphins includes the five dolphins *DN63*, *Knit*, *Oscar*, *PL*, and *SN89* that Lancichinetti *et al.* (2009) identify as members of both groups. Their method is a greedy algorithm, where in an outer loop a single uncovered vertex is randomly chosen and in an inner loop a cluster containing this vertex is determined by maximizing a fitness function.

Finally, we reduced the maximum distance to $s = 4$. The result is four overlapping clusters as depicted in Figure 7(c). Note that no vertex belongs to all four clusters, i.e., vertices are either monochrom, bicolored, or three-colored. Interestingly, Girvan and Newman (2002) also find four communities with their algorithm. Lusseau and Newman (2004) argue that Girvan and Newman (2002) found a natural decomposition of the larger community (green vertices in Figure 7(a)) into three sub-communities, where this subdivision is correlated with the gender and age of the dolphins. In comparison, our depicted decomposition consists of slightly larger clusters, but reflects well that three sub-communities can be identified in the larger community.

### *6.3.3. Football.*

Girvan and Newman (2002) introduced another social network in which the vertices are American Football college teams and edges represent regular-season games between them. The 115 teams are divided into eleven conferences containing between six and 13 teams each. Generally, teams play more intraconference than interconference games so that conferences form clusters. Moreover, there are eight independent teams

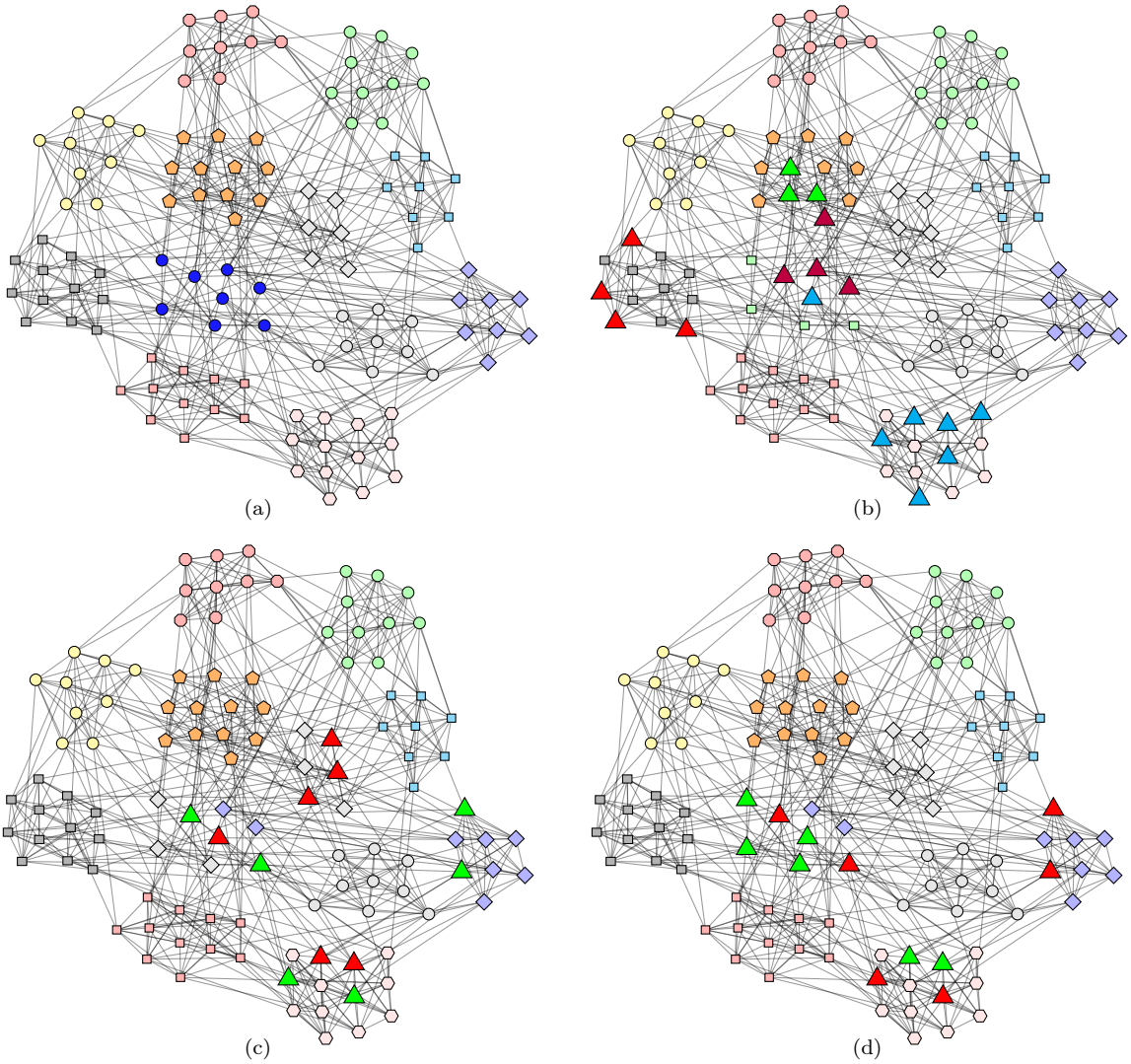


Figure 8: Football (a) Eleven Conferences and Independent Teams (blue •), (b) Partitioning with 3-plexes, 16 clusters, (c) Partitioning with 4-plexes, 13 clusters, (d) Partitioning with connected 4-plexes, 13 clusters

that do not belong to a specific conference. Their game schedule is less structured than for the conference teams meaning that games among independent teams are as likely as games between independent teams and conference teams. Overall, the interconference games are not uniformly distributed because games between geographically close teams are more frequent. The football network is depicted in Figure 8(a).

It is important to mention that the instance as provided on Marc Newman's webpage (`http://www-personal.umich.edu/~mejn/netdata/`) incorrectly assigns seven teams to conferences. To be precise, *Boise State* and *Utah State* belong to the conference *Sun Belt (Big West)*, *Texas Christian* belongs to the conference *Western Athletic*, and *Louisiana Tech*, *Louisiana Monroe*, *Middle Tennessee State*, and *Louisiana Lafayette* are independent teams. We used `https://en.wikipedia.org/wiki/2000_NCAA_Division_I-A_football_season` and `http://www.phys.utk.edu/sorensen/cfr/cfr/Output/2000/CF_2000_Main.html` as independent sources. The consequence is that several works base their presentation on an incorrect reference solution (e.g., Girvan and Newman, 2002; Zhou, 2003a,b). However, the 613 games as given by Marc Newman seem to reflect the schedule of the 2000 season.

The hierarchical decomposition methods used by Girvan and Newman (2002) and Zhou (2003a,b) provide

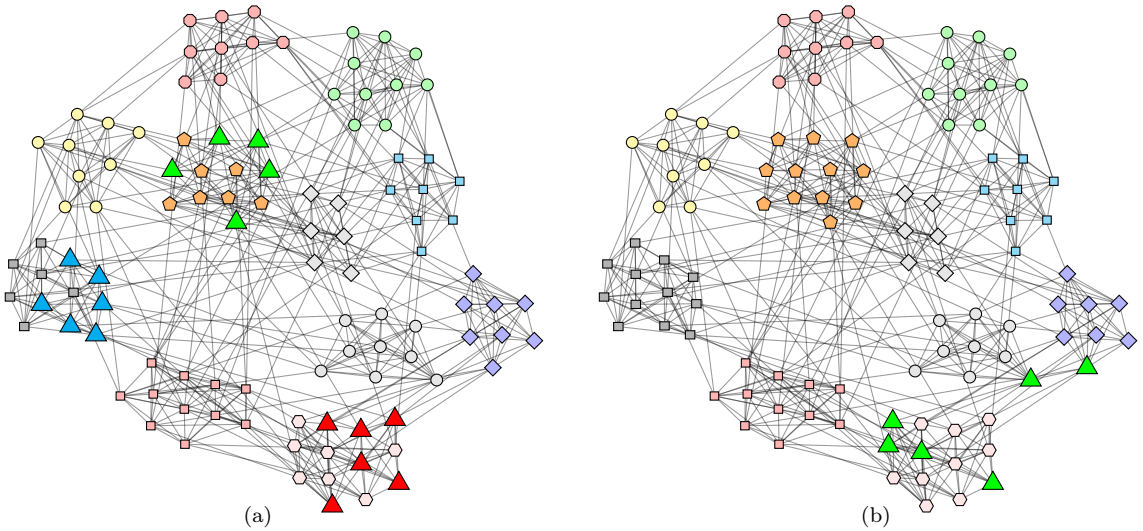


Figure 9: Football without Independent Teams (a) Partitioning into connected 3-plexes, 14 clusters (b) Partitioning in connected 4-plexes, twelve clusters

many possible clusterings but do not answer the question what the "best" number of clusters is. Later, with the definition of modularity, Newman and Girvan (2004) made it possible to assess the quality of different decompositions. With the objective of modularity maximization, Aloise *et al.* (2010) find that a clustering into ten groups is optimal. Unfortunately, they do not provide the actual solution.

For decomposing this network into relaxed cliques, we expect that conferences are well represented by $s$-plexes because games within the same conference are predominant. More precisely, with the corrected conference assignment, there are 414 intraconference games (including 10 games among the independent teams) and 199 interconference games. On average, there are ten teams per conference and each team plays seven games within its conference. Thus, the average conference constitutes a 3-plex ($s = 10 - 7$). Due to the above mentioned irregularities, also larger values of $s$ make sense.

We start our analysis of the `football` network with a partitioning into 3-plexes. Figure 8(b) shows that in this case the minimum number of partitions is 16. While eight of the eleven conferences are detected, the remaining three are structured into two or three groups that also contain some of the independent teams. The split conferences are the three largest conferences which are actually subdivided into two divisions of six or seven teams each. We later see that good decompositions can uncover this type of substructure.

In order to better meet the correct number of conferences, we decomposed the network using 4-plexes. The partitioning into 4-plexes is depicted in Figure 8(c). Also here three conferences are mixed with independent teams. However, only nine conference teams are misclassified compared to 14 conference teams in the 3-plex solution. Even with this improvement, the solution has the defect that one cluster is disconnected (teams depicted as red triangles). We therefore impose connectivity. The resulting partitioning into connected 4-plexes is shown in Figure 8(d). The number of clusters does not increase compared to the disconnected solution (13 groups). Now, one more conference is correctly detected, only two conferences are mixed with the independent teams.

We briefly mention that partitioning with 5-plexes does not decrease the number of partitions. Therefore, we omit the presentation of these results.

In the three solutions given in Figure 8(b)–(d), the independent teams are assigned very differently. This may be an indication that the independent teams do not form a community encoded by the graph. In a series of additional experiments, we therefore removed the independent teams from the network. The new network consists of 107 vertices and 551 edges, see Figure 8(a) with the independent teams (dark blue circles) removed. For the sake of brevity, we omit the explicit depiction of the new network.

We present the partitioning with connected 3-plexes and connected 4-plexes in Figure 9(a) and (b). The

3-plex partitioning consists of 14 clusters. Eight of them perfectly reproduce the smaller conferences, while three pairs of the remaining six clusters exactly form the three largest conferences. Recall that these three conferences do have subdivisions in reality. The 4-plex partitioning identifies twelve groups, one more than there are conferences. However, this is the best solution in the sense that only six teams are misclassified. The conference with the most mismatches is *Mid American* (depicted with rose hexagons). It is the only conference with 13 teams and it does not form an s-plex for $s \leq 5$ because four teams have a degree of seven. Thus, the real conferences do not form a feasible partitioning into 5-plexes. If we run our algorithm for partitioning with 5-plexes, the solution perfectly matches the correct number of eleven conferences, but groups teams of four conferences incorrectly.

## 7. Conclusions

In this paper, we have introduced the problem of decomposing a graph into a minimum number of relaxed cliques as a new method for community detection. While in prior work the resulting clusters generally do not have any structure, the different clique relaxations allow to impose application-specific constraints a cluster has to fulfill. Using the eight types of first-order clique relaxations as defined by Pattillo *et al.* (2013a), we identified 16 new relevant types of decompositions. In particular, for non-hereditary relaxed cliques one must distinguish between partitioning and covering the network. Moreover, since a basic requirement for communities is connectivity, we have introduced the concept of connected relaxed cliques. As a consequence, decomposing into connected or general relaxed cliques gives rise to different problem variants. Our type of approach is useful in cases where one has a good understanding of what defines a community. For three prominent examples from social network analysis, we have demonstrated that decomposition into relaxed cliques reproduces some known features of the network.

Modularity maximization is the predominant method in community detection to assess the quality of a clustering. Our decomposition approach is independent of modularity and might be a valid alternative to overcome the limitations of modularity maximization as discussed by Fortunato and Barthélemy (2007). They prove that modularity maximization can incorrectly identify clusters in some cases. For example, for a network composed of the union of sufficiently large cliques $K_n$ arranged in a cycle, modularity maximization joins pairs of $K_n$. Our (relaxed) clique partitioning approach would correctly identify each $K_n$ as a single cluster.

From an optimization point of view, decomposing into relaxed cliques is a hard problem. Our exact solution approach is based on branch-and-price, where pricing requires the design of new effective algorithms and branching the development of complete and preferably structure-preserving branching rules. For pricing, we propose a new CB&B for $s$-club, a modified version of RDS for hereditary relaxed cliques that is able to handle connectivity and negative weights, and new MIP models for $s$-bundle and $k$-block. For branching, a comparison of different branching schemes revealed that Ryan-Foster branching is superior although it is not structure preserving for the pricing problem.

We see several avenues for future research: For all variants, the pricing problem is a maximum-weight relaxed clique problem which is $\mathcal{NP}$-hard and also practically challenging. Effective (meta)heuristics for the solution of these problems are not available, but would certainly accelerate the column-generation process. Moreover, large-scale networks require heuristics and metaheuristics for the overall decomposition. The new column-generation algorithms analyzed here can provide tight lower bounds for assessing the metaheuristics' performance.

There is also room for alternative structures that define the clusters, e.g., additional types of relaxed cliques such as second-order relaxed cliques and $k$-connected/$k$-hereditary relaxed cliques (see Pattillo *et al.*, 2013a). Alternatively, two or more different types of relaxed cliques can be allowed meaning that a cluster can, e.g., either be a 2-plex or a 5-defective clique. Moreover, new relaxed clique definitions result when additional attributes are associated with vertices or edges. An example is a distance-$d$-clique defined as a set of vertices with a pairwise distance no exceeding $d$ (measured by the sum of edge distances $d_{ij}$). Also, the overall objective of minimizing the number of clusters can be replaced by one in which the clusters receive a weight, e.g., computed as function (maximum, sum, average, or product) of its vertex and edge weights.

## References


Abello, J., Pardalos, P., and Resende, M. G. C. (1999). On maximum clique problems in very large graphs. In J. M. Abello and J. S. Vitter, editors, *External Memory Algorithms and Visualization. DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, Boston, MA.

Almeida, M. T. and Carvalho, F. D. (2012). Integer models and upper bounds for the 3-club problem. *Networks*, **60**(3), 155–166.

Almeida, M. T. and Carvalho, F. D. (2013). An analytical comparison of the LP relaxations of integer models for the $k$-club problem. *European Journal of Operational Research*, **232**(3), 489–498.

Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., and Liberti, L. (2010). Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, **82**(4), 046112.

Balasundaram, B., Butenko, S., and Hicks, I. V. (2011). Clique relaxations in social network analysis: The maximum $k$-plex problem. *Operations Research*, **59**(1), 133–142.

Bourjolly, J.-M., Laporte, G., and Pesant, G. (2000). Heuristics for finding $k$-clubs in an undirected graph. *Computers & Operations Research*, **27**(6), 559–569.

Bourjolly, J.-M., Laporte, G., and Pesant, G. (2002). An exact algorithm for the maximum $k$-club problem in an undirected graph. *European Journal of Operational Research*, **138**(1), 21–28.

Brandes, U. and Erlebach, T., editors (2005). *Network Analysis: Methodological Foundations [outcome of a Dagstuhl seminar, 13-16 April 2004]*, volume 3418 of *Lecture Notes in Computer Science*. Springer.

Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., and Schulz, C. (2013). Recent advances in graph partitioning. *CoRR*, **abs/1311.3144**.

Carraghan, R. and Pardalos, P. M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, **9**(6), 375–382.

Carvalho, F. D. and Almeida, M. T. (2011). Upper bounds and heuristics for the 2-club problem. *European Journal of Operational Research*, **210**(3), 489–494.

Cook, V. J., Sun, S. J., Tapia, J., Muth, S. Q., Argüello, D. F., Lewis, B. L., Rothenberg, R. B., and McElroy, P. D. (2007). Transmission network analysis in tuberculosis contact investigations. *Journal of Infectious Diseases*, **196**(10), 1517–1527.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, **486**(3–5), 75–174.

Fortunato, S. and Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, **104**(1), 36–41.

Garey, M. and Johnson, D. (1979). *Computers and Intractability*. Freeman, New York.

Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, **99**(12), 7821–7826.

Gschwind, T., Irnich, S., and Podlinski, I. (2015). Maximum weight relaxed cliques and Russian doll search revisited. Technical Report LM-2015-02, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Guo, J., Komusiewicz, C., Niedermeier, R., and Uhlmann, J. (2010). A more relaxed model for graph-based data clustering: $s$-plex cluster editing. *SIAM Journal on Discrete Mathematics*, **24**(4), 1662–1683.

Held, S., Cook, W., and Sewell, E. C. (2012). Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, **4**(4), 363–381.

Kammer, F. and Täubig, H. (2005). Connectivity. In Brandes and Erlebach (2005), pages 143–177.

Kosub, S. (2004). Local density. In Brandes and Erlebach (2005), pages 112–142.

Lancichinetti, A., Fortunato, S., and Kertész, J. (2009). Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, **11**(3), 033015.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Lusseau, D. (2003). The emergent properties of a dolphin social network. *Proceedings of the Royal Society B: Biological Sciences*, **270**(Suppl_2), S186–S188.

Lusseau, D. and Newman, M. E. J. (2004). Identifying the role that animals play in their social networks. *Proceedings of the Royal Society B: Biological Sciences*, **271**(Suppl_6), S477–S481.

Mahdavi Pajouh, F. and Balasundaram, B. (2012). On inclusionwise maximal and maximum cardinality $k$-clubs in graphs. *Discrete Optimization*, **9**(2), 84–97.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, **10**(1), 147–175.

Mehrotra, A. and Trick, M. (1998). Cliques and clustering: A combinatorial approach. *Operations Research Letters*, **22**, 1–12.

Menger, K. (1927). Zur allgemeinen Kurventheorie. *Fund. Math.*, **10**, 96–115.

Nemhauser, G. and Park, S. (1991). A polyhedral approach to edge coloring. *Operations Research Letters*, **10**, 315–322.

Nemhauser, G. and Trotter Jr., L. (1974). Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, **6**, 48–61.

Nemhauser, G. and Trotter Jr., L. (1975). Vertex packings: Structural properties and algorithms. *Mathematical Programming*, **8**(1), 232–248.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, **69**, 026113.

Östergård, P. R. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, **120**(1-3), 197–207.

Pattillo, J., Youssef, N., and Butenko, S. (2012). Clique relaxation models in social network analysis. In M. T. Thai and P. M.

Pardalos, editors, *Handbook of Optimization in Complex Networks*, volume 58 of *Springer Optimization and Its Applications*, pages 143–162. Springer New York.

Pattillo, J., Youssef, N., and Butenko, S. (2013a). On clique relaxation models in network analysis. *European Journal of Operational Research*, **226**(1), 9–18.

Pattillo, J., Veremyev, A., Butenko, S., and Boginski, V. (2013b). On the maximum quasi-clique problem. *Discrete Applied Mathematics*, **161**(1–2), 244–257.

Porter, M. A., Onnela, J.-P., and Mucha, P. J. (2009). Communities in networks. *Notices of the AMS*, **56**(9), 1082–1097.

Ryan, D. and Foster, B. (1981). An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, chapter 17, pages 269–280. Elsevier, North-Holland.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, **1**(1), 27–64.

Scott, J. (2012). *Social Network Analysis*. Sage, London, UK, 3rd edition edition.

Shahinpour, S. and Butenko, S. (2013). Algorithms for the maximum $k$-club problem in graphs. *Journal of Combinatorial Optimization*, **26**(3), 520–554.

Sherali, H. D. and Smith, J. C. (2006). A polyhedral study of the generalized vertex packing problem. *Mathematical Programming*, **107**(3), 367–390.

Trukhanov, S., Balasubramaniam, C., Balasundaram, B., and Butenko, S. (2013). Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations. *Computational Optimization and Applications*, **56**(1), 113–130.

Vanderbeck, F. (2011). Branching in branch-and-price: a generic scheme. *Mathematical Programming*, **130**(2), 249–294.

Veremyev, A. and Boginski, V. (2012). Identifying large robust network clusters via new compact formulations of maximum $k$-club problems. *European Journal of Operational Research*, **218**(2), 316–326.

Verfaillie, G., Lemaître, M., and Schiex, T. (1996). Russian doll search for solving constraint optimization problems. In *Proceedings of the thirteenth national conference on Artificial intelligence*, volume 1, pages 181–187. AAAI Press.

Wasserman, S. and Faust, K. (1994). *Social Network Analysis*. Cambridge University Press.

Wotzlaw, A. (2014). On solving the maximum $k$-club problem. Technical Report arXiv:1403.5111v2, Institut für Informatik, Universität zu Köln, Köln, Germany.

Yannakakis, M. (1978). Node-and edge-deletion NP-complete problems. In *STOC '78: Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–264, New York, NY. ACM Press.

Yu, H., Paccanaro, A., Trifonov, V., and Gerstein, M. (2006). Predicting interactions in protein networks by completing defective cliques. *Bioinformatics*, **22**(7), 823–829.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, **33**(4), 452–473.

Zhou, H. (2003a). Distance, dissimilarity index, and network community structure. *Physical Review E*, **67**(6), 061901.

Zhou, H. (2003b). Network landscape from a Brownian particle's perspective. *Physical Review E*, **67**(4), 041908.

## Appendix

### A. MIP Formulation for Maximum-Weight $s$-Club

We assume that the simple graph $G = (V, E)$ with vertex weights $w_i, i \in V$ is given together with some $s \geq 2$.

Veremyev and Boginski (2012) proposed the first compact formulation for the maximum cardinality $s$-club problem, i.e., a formulation with a polynomial number of variables and constraints (polynomial in $s$, $|V|$, and $|E|$). In addition to vertex variables $x_i \in \{0, 1\}$ defining $S = \{i \in V : x_i = 1\}$, there are variables $v_{ij}^{\ell} \in \{0, 1\}$ indicating that an $i$-$j$-path of length $\leq \ell$ exists in $G[S]$, i.e., $d_{G[S]}(i, j) \leq \ell$. In (Veremyev and Boginski, 2012), the domain of the indices $\ell$ is not completely defined. We therefore present a sightly modified version of the model in which a minimum number of the path variables $v_{ij}^{\ell}$ is needed. Since $i$-$j$-paths of length one are the edges $\{i, j\}$, the domain of the index $\ell$ can be defined as $dom_2(i, j) = \{\max\{2, d_G(i, j)\}, \ldots, s\}$. Similarly, we define $dom_3(i, j) = \{\max\{3, d_G(i, j)\}, \ldots, s\}$. With the definition $\mathcal{U} = \{\{i, j\} : i, j \in V, i < j\}$

for unordered pairs, the formulation of the maximum weight $s$-club problem is:

$$\max \quad \sum_{i \in V} w_i x_i \tag{5a}$$

$$\text{s.t.} \quad v_{ij}^{\ell} \le x_i, \quad v_{ij}^{\ell} \le x_j \qquad \{i,j\} \in \mathcal{U}, \ell \in dom_2(i,j) \tag{5b}$$

$$\sum_{\ell \in dom_2(i,j)} v_{ij}^{\ell} \ge x_i + x_j - 1 \qquad \{i,j\} \in \mathcal{U} \setminus E \tag{5c}$$

$$v_{ij}^{2} \le \sum_{p \in N(i) \cap N(j)} x_p \qquad \{i,j\} \in \mathcal{U}, \text{dist}_G(i,j) = 2 \tag{5d}$$

$$v_{ij}^{\ell} \le \sum_{p \in N(i), \text{dist}(p,j) \le \ell - 1} v_{pj}^{\ell - 1} \qquad \{i,j\} \in \mathcal{U}, 2 \le dist_G(i,j) \le s, \ell \in dom_3(i,j) \tag{5e}$$

$$x_i \in \{0,1\} \qquad i \in V \tag{5f}$$

$$v_{ij}^{\ell} \in \{0,1\} \qquad \{i,j\} \in \mathcal{U}, \ell \in dom_2(i,j) \tag{5g}$$

Due to (5b), the selection of a path associated with $v_{ij}^{\ell}$ is only possible if both endpoints are present. Conversely, the constraints (5c) allow the selection of both vertices $i$ and $j$ if and only if there exists a path of length $\le s$ between them in $G[S]$. The constraints (5d) and (5e) model the construction of paths in $G[S]$ connecting $i$ and $j$. The first constraints guarantee that a vertex adjacent to $i$ and $j$ is selected for the distance two, while the latter work recursively. A path of length $\ell$ between vertices $i$ and $j$ requires the selection of a vertex $p$ adjacent to $i$ together with the presence of another path of length $\ell - 1$ between $p$ and $j$. The domains of the vertex and path variables are defined by (5f) and (5g). Note that clique-like constraints $x_i + x_j \le 1$ for incompatible vertices $i, j \in V$ are present in the above formulation: If $d_G(i,j) > s$ for $i, j \in V$, no $s$-club can contain both vertices, and $dom_2(i,j)$ is the empty set by definition so that the corresponding constraint (5c) reduces to $x_i + x_j \le 1$. Hence, any valid inequalities for the clique polytope of the corresponding power graph $G^s$ are valid and may be used to strengthen the LP relaxation of the model. Moreover, Veremyev and Boginski (2012) presented additional valid inequalities for $\mathscr{F}(G)$, but in our computational test on maximum cardinality and maximum weight $s$-club (pricing) problems they did not improve the performance.

### B. MIP Formulation for Maximum-Weight $s$-Bundle and $k$-Block

We assume that the simple graph $G = (V, E)$ with vertex weights $w_i, i \in V$ is given together with some $s \ge 2$. We now present a compact MIP formulation for the maximum-weight $s$-bundle problem.

Let $\mathcal{N} = (N, A)$ be the auxiliary network associated with $G$ defined as follows: For each vertex $i \in V$ there exist two vertices $i^-$ and $i^+$ in $\mathcal{N}$ so that $N = V^+ \cup V^-$. The network $\mathcal{N}$ comprises two types of arcs. First, for all $i \in V$, arcs $(i^-, i^+)$ are present in $A$. Second, for each edge $\{i,j\} \in E$, the arcs $(i^+, j^-)$ and $(j^+, i^-)$ are in $A$. Hence, $A = \{(i^-, i^+) : i \in V\} \cup \{(i^+, j^-), (j^+, i^-) : \{i,j\} \in E\}$. Now, any two non-adjacent vertices $i, j \in V$ are $k$-connected in $G$ if and only if there exists a flow of value $k$ between $i^+$ and $j^-$ in $\mathcal{N}$. (All arcs have unit capacity.) The same holds for $G[S]$ and the induced network $\mathcal{N}[S^+ \cup S^-]$ for any $S \subseteq V$.

Three types of decision variables are in the MIP: The binary variables $x_i, i \in V$ indicate whether or not vertex $i \in V$ is in the selected $s$-bundle $S = \{i \in V : x_i = 1\}$. The continuous variable $z$ describes the number $|S| - s$ of vertex-disjoint paths that must exists between non-adjacent pairs of vertices of $S$. With the definition $\mathcal{U} = \{\{i,j\} : i, j \in V, i < j\}$ for unordered pairs, for each $\{i,j\} \in \mathcal{U} \setminus E$, the binary variables

$y_a^{ij}, a \in A$ model flows in $\mathcal{N}$ connecting $i^+$ and $j^-$.

$$\max \quad \sum_{i \in V} w_i x_i \tag{6a}$$

$$\text{s.t.} \quad z \geq \sum_{i \in V} x_i - s \tag{6b}$$

$$\sum_{a \in \delta^+(i^+)} y_a^{ij} \geq z - M^{ij}(2 - x_i - x_j) \qquad \{i,j\} \in \mathcal{U} \setminus E \tag{6c}$$

$$\sum_{a \in \delta^+(n)} y_a^{ij} - \sum_{a \in \delta^-(n)} y_a^{ij} = 0 \qquad \{i,j\} \in \mathcal{U} \setminus E, n \in N, n \neq i^+, j^- \tag{6d}$$

$$\sum_{a \in \delta^-(j^-)} y_a^{ij} \geq z - M^{ij}(2 - x_i - x_j) \qquad \{i,j\} \in \mathcal{U} \setminus E \tag{6e}$$

$$y_{p^- p^+}^{ij} \leq x_p \qquad \{i,j\} \in \mathcal{U} \setminus E, p \in V \tag{6f}$$

$$x_i \in \{0,1\} \qquad i \in V \tag{6g}$$

$$y_a^{ij} \geq 0 \qquad a \in A, \{i,j\} \in \mathcal{U} \setminus E \tag{6h}$$

$$z \geq 0 \tag{6i}$$

The objective (6a) maximizes the sum of the vertex weights in the selected $s$-bundle $S$. The constraint (6b) guarantees $z \geq |S| - s$. The next three groups of constraints (6c)–(6e) ensure a flow of at least $z$ between $i^+$ and $j^-$ in case that $i$ and $j$ belong to the bundle. Herein, $M^{ij} > 0$ is a sufficiently large number. The coupling constraints (6f) guarantee that flows are positive only in $\mathcal{N}[S^+ \cup S^-]$. The domains of all variables are stated in (6g)–(6i).

In an $s$-bundle $S$, every vertex must have a degree $\deg_{G[S]}(i)$ not smaller than $|S| - s$ (see Pattillo et al., 2013b, p. 17). Based on this observation, we can find a feasible, but small value for $M^{ij}$ in constraints (6c) and (6e) in order to tighten the formulation:

$$M^{ij} := \max\{k \in \mathbb{N} : \exists S \subseteq V, |S| = k, \forall v \in S : \max\{\deg_{G \setminus \{i\}}(v), \deg_{G \setminus \{j\}}(v)\} \geq k\}$$

This maximum can be computed by simply sorting all vertices decreasingly by the values $\max\{\deg_{G \setminus \{i\}}(v), \deg_{G \setminus \{j\}}(v)\}$.

Note that a similar formulation can be used to find maximum-weight $k$-blocks. The variable $z$ can be replaced by the constant $k$ so that (6b) and (6i) are obsolete.

### C. Handling Connectivity Constraints and Negative Weights in RDS

Recall from Section 5.1 that the Russian doll search (RDS) is an algorithm for finding maximum weight relaxed cliques, which are defined by a hereditary property $\Pi$. We now present the necessary modifications that allow us to find *connected* relaxed cliques. The resulting structure of a connected relaxed clique is no longer hereditary, which implies that negative vertex weights generate additional complications. Such negative vertex weights result form three facts: (i) the weights (before branching) are equal to the dual prices of the constraints (4b) which can be negative in case of partitioning, (ii) the implementation of separate-constraints imposes large negative weights on the vertices (see Section 5.3), and (iii) dual prices of constraints that bound the number of vertex contacts from above are non-positive summands of the weights (see Section 5.2.3). The modified RDS is applicable to $s$-clique, $s$-plex, $s$-defective clique, and $s$-bundle. Our description of RDS follows the presentation of Trukhanov et al. (2013) and our own work (Gschwind et al., 2015).

In standard RDS (Algorithm 2), the $n$ of vertices $V$ are ordered into a sequence $(v_1, v_2, \ldots, v_n)$. Instead of one depth-first branch-and-bound search, $n$ searches are performed in the main loop of RDS (Steps 3-6 of Algorithm 3). Starting from $i = n$, the $i$th search determines a maximum weight $\Pi$ set for $G[\{v_i, v_{i+1}, \ldots, v_n\}]$

**Algorithm 2:** RDS for the Maximum Weight $\Pi$ Problem

**Input**: $G = (V, E, w_i)$ with $w_i \in \mathbb{R}_0^+$; $\Pi$

Order vertices $(v_1, v_2, \ldots, v_n)$
Set $LB := 0$ and $S := \emptyset$
**for** $i := n, n-1, \ldots, 1$ **do**
    Set $C := \{v_j : j > i, \{v_i, v_j\} \text{ satisfies } \Pi\}$
    Call FindMax($C$, $\{v_i\}$)
    $LB_i := LB$

**Output**: $S \subseteq V$ inducing a maximum weight $\Pi$ subgraph $G[S]$

**Algorithm 3:** Modified RDS for the Maximum Weight Connected $\Pi$ Problem

**Input**: $G = (V, E, w_i)$ with $w_i \in \mathbb{R}$, $\Pi$

Order vertices $(v_1, v_2, \ldots, v_n)$
Set $LB := 0$, $LB^c := 0$, and $S := \emptyset$
**for** $i := n, n-1, \ldots, 1$ **do**
    Set $C := \{v_j : j > i, \{v_i, v_j\} \text{ satisfies } \Pi\}$
    Call FindMaxConnected($C$, $\{v_i\}$)
    $LB_i := LB$

**Output**: $S \subseteq V$ inducing a maximum weight connected $\Pi$ subgraph $G[S]$

**Procedure** FindMax($C$, $P$)

**Input**: Candidate set $C$, current set $P$

**if** $C = \emptyset$ **then**
    **if** $w(P) > LB$ **then**
        Set $LB := w(P)$ and $S := P$
    **return**
**while** $C \neq \emptyset$ **do**
    **if** $w(C) + w(P) \leq LB$ **then return**
    Set $i := \min\{j : v_j \in C\}$
    **if** $LB_i + w(P) \leq LB$ **then return**
    Set $C := C \setminus \{v_i\}$ and $P' := P \cup \{v_i\}$
    PrepareAuxiliaryInformation($C$, $P'$)
    Set $C' := \{v \in C : P' \cup \{v\} \text{ satisfies } \Pi\}$
    Call FindMax($C'$, $P'$)

**Procedure** FindMaxConnected($C$, $P$)

**Input**: Candidate set $C$, current set $P$

**if** $C = \emptyset$ **then**
    **if** $w(P) > LB$ **then**
        Set $LB := w(P)$
    Set $P^* := \underset{R \text{ connected comp. of } G[P]}{\operatorname{argmax}} \sum_{r \in R} w_r$
    **if** $w(P^*) > LB^c$ **then**
        Set $LB^c := w(P^*)$ and $S := P^*$
    **return**
**while** $C \neq \emptyset$ **do**
    **if** $w^+(C) + w(P) \leq LB^c$ **then return**
    Set $i := \min\{j : v_j \in C\}$
    **if** $LB_i + w(P) \leq LB^c$ **then return**
    Set $C := C \setminus \{v_i\}$ and $P' := P \cup \{v_i\}$
    PrepareAuxiliaryInformation($C$, $P'$)
    Set $C' := \{v \in C : P' \cup \{v\} \text{ satisfies } \Pi\}$
    Call FindMaxConnected($C'$, $P'$)
Call FindMaxConnected($\emptyset$, $P$)

with the initial set $S = \{v_i\}$ by means of the recursion FindMax. In every iteration, $i$ is decreased by 1 so that a sequence of lower bounds $LB_n, LB_{n-1}, \ldots, LB_2, LB_1$ is computed, where $LB_i$ corresponds to the value of a maximum weight $S \subseteq \{v_i, \ldots, v_n\}$ fulfilling $\Pi$. The value of the best solution found so far is retained in the overall lower bound $LB$. At each stage of the RDS search, the current solution $P$ satisfies $\Pi$. Moreover, a set of candidates $C$ with $P \cup \{c\}$ satisfies $\Pi$ for all $c \in C$ is maintained. Whenever $P$ is enlarged, $C$ has to be adjusted, i.e., candidate vertices not compatible with the new set $P$ are removed from $C$.

The positive part of a number $w$ is denoted by $w^+ = \max\{0, w\}$. We use the shorthand notation $w(P) = \sum_{v \in P} w_v$ and $w^+(P) = \sum_{v \in P} w_v^+$.

The modified RDS (Algorithm 3) uses two types of overall lower bounds instead of just one: $LB$ is, as in the standard RDS, the maximum weight of subset $P \subseteq V$ fulfilling $\Pi$ either connected or not. $LB^c$ is the maximum weight of subset $P \subseteq V$ fulfilling $\Pi$ that is connected.

Furthermore, the pruning criteria of the standard RDS and the modified RDS differ. The standard weight-based pruning ($w(C)+w(P) \leq LB$, Step 6 in Procedure FindMax) is adapted so that it takes possibly negative weights into account and compares against the connected bound, i.e., $w^+(C)+w(P) \leq LB^c$ in Step 9 of Procedure FindMaxConnected. The RDS-specific pruning Step 11 of Procedure FindMaxConnected compares $LB_i + w(P)$ against $LB^c$ instead of $LB$. Note that $LB_i$ is the maximum weight of a general (connected or not) $P \subseteq \{v_i, \ldots, v_n\}$ fulfilling $\Pi$. Both modified pruning steps are less effective compared to the pruning steps of the standard RDS.

When the candidate set $C$ becomes empty, the modified RDS does three things: First, the lower bound $LB$ is updated whenever $P$ is improving (Steps 2 and 3 in Procedures FindMax and FindMaxConnected). Second, the connected components $R$ of $G[P]$ are computed. For $s$-plex, $s$-defective clique, and $s$-bundle, this step is obsolete whenever $|P|$ is sufficiently large (see Table 1), while for smaller $|P|$ we use a straightforward enumeration. For $s$-clique, the connected components are determined with an efficient union-find algorithm (see Cormen et al., 2009, § 21.3). In all cases, the component $P^*$ with largest weight $w(P^*)$ is determined. Third, if $w(P^*)$ improves the connected lower bound $LB^c$, a new improving connected relaxed clique is found and $LB^c$ as well as $S$ are updated.

Finally, the additional Step 16 of FindMaxConnected is required, since all candidates $C$ may have a negative weight so that the current set $P$ without any vertex additions has the largest weight among all subsets $S$ with $P \subseteq S \subseteq P \cup C$.

Note that it is sufficient to consider only the connected components of $G[P]$ instead of all connected subsets of $P$ in Steps 4-6 of the recursion. The reason is that for given sets $P$ and $C$ the RDS enumerates all subsets of $P \cup C$ as long as no pruning occurs. Indeed, the modified pruning Step 11 guarantees that no subset of $P \cup C$ fulfilling $\Pi$ (connected or not) and therefore no connected subset of $P \cup C$ fulfilling $\Pi$ is excluded. Thus, connected subsets of $P$ that are not connected components of $G[P]$ are found as connected components of the induced subgraph of a different current set $P'$ in another iteration of Procedure FindMaxConnected.

### D. New Combinatorial Branch-and-Bound for Maximum Weight $s$-Club

We developed a new combinatorial branch-and-bound algorithm for maximum weight $s$-club which is able to handle arbitrary vertex weights $w_i \in \mathbb{R}$. Since $s$-club is non-hereditary, we have to cope with negative weights. As before, we assume that the simple graph $G = (V, E)$ with vertex weights $w_i, i \in V$ is given together with some $s \geq 2$.

Throughout the branch-and-bound, we partition the vertices $V$ into three sets, the *included vertices* $I$ with $x_i = 1$ for all $i \in I$, the *free vertices* $F$ with $x_i \in \{0, 1\}$ for all $i \in F$, and the *excluded vertices* $X$ with $x_i = 0$ for all $i \in X$. The algorithm is initialized with $F = V$ and $I = X = \emptyset$. All partitions with $F \neq \emptyset$ are partial solutions, while those with $F = \emptyset$ are complete solutions. We always assume that the partition $V = I + F + X$ must admit that a subset $S$ of the *admissible vertices* $I \cup F$ is a feasible $s$-club with $S \supseteq I$. In particular, the distance between all admissible vertices $i \in F \cup I$ and the included vertices $j \in I$ must not exceed $s$, i.e., $\text{dist}_{G[I \cup F]}(i, j) \leq s$, meaning that $I \cup \{j\}$ is an $s$-clique for all $j \in F$. This is similar to the branch-and-bound algorithm by Mahdavi Pajouh and Balasundaram (2012). However, our approach differs

in the computation of upper and lower bounds and it has an additional component for fixing vertices and detecting infeasible partial solutions.

*Upper Bounds..* A straightforward upper bound is

$$ub_1(I+F) = \pi(I) + \pi^+(F) = \sum_{i \in I} w_i + \sum_{i \in F} w_i^+$$

with the standard shorthand notation $w_i^+ := \max\{0, w_i\}$ for referring to the positive part.

In the course of the algorithm, the induced graph $G[I \cup F]$ can become disconnected with components $C_1, \ldots, C_p$. Since every $s$-club is connected, the component with the largest weight provides a tighter bound in this case:

$$ub_2(I+F) = \min_{j=1,\ldots,p} ub_1(C_j).$$

A third upper bound results from the fact that every $s$-club is also an $s$-clique. This is a (1-)clique the $s$th power graph $G^s$. Therefore,

$$ub_3(I+F) = w(I) + z(w, G^s[F]),$$

where $z(w, G^s[F])$ is the maximum weight of a clique that can be found in the induced subgraph by $F$ in $G^s$. Any algorithm for computing maximum weight cliques can be used to compute $z(w, G^s[F])$ (or any algorithm for maximum weight independent sets in the complement graph of $G^s$). We used the publicly available code from the paper (Held et al., 2012).

*Lower Bounds..* For computing an initial lower bound, we adapt the `Constellation` heuristic of Bourjolly et al. (2002). `Constellation` first determines a start vertex $i$ with maximum value $w_i + \sum_{j \in N(i)} w_j^+$. The start solution is $S = \{i\} \cup \{j \in N(i) : w_j \geq 0\}$. Then, $s-2$ times additional vertices are added. For all vertices $j \in S$, one with maximum $\sum_{i \in N(j) \setminus S} w_i^+$ is determined and all vertices $N(j) \setminus S$ are added to $S$. Note that this step increases the diameter of $G[S]$ by at most one so that after $s-2$ iterations the diameter of $S$ cannot exceed $s$, i.e., $G[S]$ is an $s$-club. Since no computation of distances in induced subgraphs is needed, the `Constellation` heuristic provides a lower bound very quickly. Preliminary test revealed that the `Drop` heuristic of Bourjolly et al. (2002) is in many cases too time consuming, and we do not use it.

The only primal heuristic that we employ at every branch-and-bound node is the check whether or not $I \cup F$ induces a feasible $s$-club. It requires the computation of a distance matrix for $G[I \cup F]$ (with unit costs), which is implemented as a straightforward breadth-first-search (BFS). It requires not more than $\mathcal{O}(n_{I \cup F} \cdot m_{I \cup F})$ time, where $n_{I \cup F} = |I \cup F|$ is the number of vertices and $m_{I \cup F}$ is the number of edges of $G[I \cup F]$.

*Vertex Fixation and Infeasibility Detection..* In the branch-and-bound algorithm, branches result from the inclusion or exclusion of a free vertex, i.e., we select a free vertex $v \in F$, remove it from $F$, and add it to either $I$ or $X$. Obviously, the inclusion of a free vertex does not change distances of $G[I \cup F]$. In contrast, the exclusion of a free vertex requires the re-computation of the distances in the graph $G[I \cup F]$ using BFS. For the sake of brevity, let $d_{ij}$ be the short notation for $\text{dist}_{G[F \cup I]}(i, j)$. The following cases are interesting:

1. If $d_{ij} > s$ for $i, j \in I$ then the partial solution is infeasible.
2. If $d_{ij} > s$ for $i \in I, j \in F$ then vertex $j$ can be excluded, i.e., $X' = X \cup \{j\}$ and $F' = F \setminus \{j\}$.
3. If $d_{ij} = s$ for $i, j \in I$ and there exists a $0 < d < s$ and a *unique* vertex $k \in F$ with $d_{ik} = d$ and $d_{kj} = s - d$, then vertex $k$ can be included, i.e., $I' = I \cup \{j\}$ and $F' = F \setminus \{j\}$.

If case 2 applies, distances need to be recalculated. If one of the cases 2 or 3 applies, the variable fixation and infeasibility detection procedure is repeated. This can create a longer sequence of fixations.

*Branching..* None of the bounding procedures can cope precisely with negative weights, and therefore we start branching by first selecting a free vertex $v \in F$ with smallest negative weight $w_v < 0$ (if any). Then, at the second level, the selection of a vertex for branching is based on the idea of choosing a highly influential free vertex $v \in F$. It is a vertex $v$ maximizing the number of included vertices to which the distance is exactly $s$. Formally, the vertex $v \in F$ maximizes $|\{i \in I : d_{iv} = s\}|$ and ties are brocken by preferring larger weights $w_v$. Finally, the overall branching strategy is depth-first, where the branch in which the free vertex is included is inspected before the branch in which the free vertex is excluded.

The overall branch-and-bound is summarized in Algorithm 4 and the recursion in Procedure Recursion.

**Algorithm 4:** Combinatorial B&B Algorithm

**Input:** Graph $G = (V, E)$, vertex weights $w = (w_i)$
SET $lb :=$ `Constellation`$(G, w)$, $ub := \infty$, $I = \emptyset$, $F = V$, and $X = \emptyset$
Recursion$(I, F, X)$
**Output:** maximum weight $s$-Club $S^*$ with weight $lb$

**Procedure** Recursion$(I, F, X)$

**if** $lb > ub_1(I \cup F)$ *or* $lb > ub_2(I \cup F)$ *or* $lb > ub_3(I \cup F)$ **then** RETURN
**if** $I \cup F$ *induces s-club* ***and*** $w(I \cup F) > lb$ **then** SET $lb := w(I \cup F)$
Select branching vertex $j \in F$
**for** *branches* $(I \cup \{j\}, F \setminus \{j\}, X)$ *and* $(I, F \setminus \{j\}, X \cup \{j\})$ **do**
    DETECT infeasibility, FIX additional vertices with result $(I', F', X')$
    **if** *infeasible* **then** RETURN
    **if** $F' = \emptyset$ **then**
        **if** $w(I') > lb$ **then** SET $lb = w(I')$ and $S^* := I'$
        RETURN
    Recursion$(I', F', X')$

We briefly compare our new algorithm with the fastest algorithms from the literature. Wotzlaw (2014) presents two MAX-SAT formulations that are solved by a state-of the art SAT solver (`clasp` 2.1.3). His comparison shows that these implementations often outperform the older algorithms presented in (Veremyev and Boginski, 2012; Mahdavi Pajouh and Balasundaram, 2012; Shahinpour and Butenko, 2013). Table 11 shows the computation times for computing a maximum-cardinality $s$-club for $s \in \{2, 3, 4\}$. For the sake of brevity, column *Other* is the shortest runtime obtained with any of the algorithms compared in (Wotzlaw, 2014), while *New* is our computation time (computed on different machines of comparable performance). It seems that our implementation is competitive, in particular, it scales well for larger instances and $s > 2$.

### E. Reduction Problem

The *reduction problem* in Step 4 of Algorithm 1 of the generalized branching rule (GBR) is the following: We are given a property $\Pi$, a graph $H = (S, E(S))$ not fulfilling $\Pi$, and a connected subgraph $(S, E')$ spanning $S$. Find a set $S^\star \subseteq S$ of minimum cardinality such that $G'[S^\star]$ is connected and $G[S^\star]$ does not fulfill $\Pi$. (Note that the connected subgraph $(S, E')$ takes the role of $G^\lambda[S]$ in GBR.)

We suspect that the reduction problem is $\mathcal{NP}$-hard for arbitrary relaxed cliques defined via $\Pi$ and, thus, propose a simple greedy procedure for its resolution. The procedure is inspired by Prim's algorithm for computing a minimum spanning tree, and it works as follows: In a first step, the vertices are sorted by increasing degree. Second, we chose a vertex $i \in S$ with smallest vertex degree and set $S^\star = \{i\}$. Iteratively, vertices $j \in S \setminus S^\star$ are tested whether or not $j$ is adjacent to $S^\star$ in $(S, E')$. The first vertex $j$, in the order of increasing degree, which fulfills the condition is added to $S^\star$. The greedy algorithm stops with the solution $S^\star$ as soon as $H[S^\star]$ does not fulfill $\Pi$.

Table 11: Computation times of $s$-club algorithms.

| Instance | $s=2$ | | $s=3$ | | $s=4$ | |
|---|---|---|---|---|---|---|
| | Other | New | Other | New | Other | New |
| `adjnoun` | 0.010 | 0.003 | 0.020 | 0.059 | 0.340 | 0.017 |
| `football` | 0.020 | 0.015 | 0.320 | 4.118 | 0.001 | 0.003 |
| `jazz` | 0.060 | 0.005 | 1.050 | 0.125 | 8.870 | 0.075 |
| `celegansneutral` | 0.430 | 1.060 | 1.060 | 1.058 | 45.300 | 0.435 |
| `email` | 16.100 | 0.037 | TL | TL | TL | TL |
| `polblogs` | 46.900 | 0.086 | TL | TL | TL | TL |

## F. Equivalence of Partitioning and Covering with a Mimimum Number of Connected $s$-Cliques

In this section, we prove the theorem stated in Section 4.

THEOREM 1. *Partitioning and covering a graph with a minimum number of connected $s$-cliques are equivalent problems for all $s \geq 1$. There exists a constructive procedure to transform a covering solution into a partitioning solution using an identical number of subsets.*

Proof: Due to the heredity and connectedness of (1-)cliques the statement is certainly true for $s = 1$. We only consider the case $s \geq 2$ in the following. We show that for any set-covering solution $S_1, S_2, \ldots, S_p$ with more than $|V|$ vertex contacts, i.e., $|S_1| + |S_2| + \cdots + |S_p| > |V|$, we can construct a new solution which has a smaller number of vertex contacts. A solution with $|V|$ vertex contacts, i.e., a set-partitioning solution, then results by iteratively applying the procedure.

We assume that the number of vertex contacts exceeds $|V|$. Then, there exists at least one vertex $v$ with at least two vertex contacts. Without loss of generality we can assume that $v \in S_1 \cap S_2$ holds. If $G[S_1 \setminus \{v\}]$ is connected, we can replace $S_1$ by $S_1 \setminus \{v\}$ in the solution, the number of vertex contacts decreases by one, and the statement follows. The same reduction can be done with $S_2$ instead of $S_1$. We can, therefore, restrict our analysis to the case that $v$ is a vertex separator of $G[S_1]$ and $G[S_2]$ (the removal of a vertex separator disconnects a graph). Let $C_1^1, C_1^2, \ldots, C_1^{p_1}$ ($p_1 \geq 2$) be the components of $G[S_1 \setminus \{v\}]$. Similarly, let $C_2^1, C_2^2, \ldots, C_2^{p_2}$ ($p_2 \geq 2$) be the components of $G[S_2 \setminus \{v\}]$.

Now we consider the maximal distance between the vertex separator $v$ and the vertices in the components. Let, $d_i^j = \max_{w \in C_i^j} \mathrm{dist}_G(v, w)$ be the maximum distance per component for all $i = 1, 2$ and $j = 1, 2, \ldots, p_i$. Since $\mathrm{dist}_G(u, w) \leq s$ for all vertices $u, w \in S_1$ (similarly $u, w \in S_2$), we know that $1 \leq d_i^j \leq s - 1$ holds for all $i = 1, 2$ and $j = 1, 2, \ldots, p_i$.

Before we consider the general case, we start with an analysis for $s = 2$: Here, the only possible value is $d_i^j = 1$ for all $i = 1, 2$ and $j = 1, 2, \ldots, p_i$. Hence, $G[S_1 \cup S_2]$ fulfills $\mathrm{dist}_{G[S_1 \cup S_2]}(i, j) \leq 2$ for any pair $i, j \in V$. We can therefore replace $S_1$ and $S_2$ by $S_1 \cup S_2$ in the solution.

For simplification, we assume from now on that

(i) the distances are sorted decreasingly, i.e., $d_1^1 \geq d_1^2 \geq \cdots \geq d_1^{p_1}$ and $d_2^1 \geq d_2^2 \geq \cdots \geq d_2^{p_2}$ and
(ii) $d_1^1 \geq d_2^1$ (otherwise one can swap $S_1$ and $S_2$).

For $s = 3$, the only possible values are $d_i^j \in \{1, 2\}$ for all $i = 1, 2$ and $j = 1, 2, \ldots, p_i$. Moreover, $d_1^1 = d_1^2 = 2$ (or $d_2^1 = d_2^2 = 2$) is impossible because otherwise $\mathrm{diam}_{G[S_1]} = 4 > s$ ($\mathrm{diam}_{G[S_2]} = 4 > s$). Hence, $d_1^2 = \cdots = d_1^{p_1} = 1$. Then, $S_2' := S_2 \cup (C_1^2 \cup \ldots \cup C_1^{p_1}) = (\{v\} \cup C_2^1 \cup C_2^2 \cup \ldots \cup C_2^{p_2}) \cup (C_1^2 \cup \ldots \cup C_1^{p_1})$ fulfills $\mathrm{diam}_{G[S_2']} = 3 \leq s$. Also $S_1' := S_1 \setminus (\{v\} \cup C_1^2 \cup \ldots \cup C_1^{p_1}) = C_1^1$ fulfills $\mathrm{diam}_{G[S_1']} \leq s$. Consequently, one can replace $S_1$ and $S_2$ by $S_1'$ and $S_2'$ in the solution. Note that $S_1' \cup S_2' = S_1 \cup S_2$ by construction, but $|S_1'| + |S_2'| = |S_1| + |S_2| - 1$ due to $v \notin S_1'$.

We can now consider the general with case $s \geq 4$: If $d_1^1 \leq s/2$ then all distances fulfill $d_i^j \leq s/2$ for all $i = 1, 2$ and $j = 1, 2, \ldots, p_i$, and one can replace $S_1$ and $S_2$ by $S_1 \cup S_2$. We have $|S_1 \cup S_2| = |S_1| + |S_2| - 1$ completing this case.

Thus, we assume $d_1^1 > s/2$. Since $\text{diam}_{G[S_1]} \leq s$, we know that

$$d_1^2, \ldots, d_1^{p_1} \leq s - d_1^1 \leq s - d_2^1 \leq s - d_2^j \quad \text{for all } j = 1, \ldots, p_2,$$

where the second inequality results from simplification (ii) and the third from simplification (i). We define $S_2' := S_2 \cup (C_1^2 \cup \ldots \cup C_1^{p_1})$ and the above inequality implies that $\text{diam}(G[S_2']) \leq s$. Moreover, we set $S_1' := C_1^1$ clearly having $\text{diam}(G[S_1']) \leq s$. As before, one can replace $S_1$ and $S_2$ by $S_1'$ and $S_2'$ in the solution. Note finally that $S_1' \cup S_2' = S_1 \cup S_2$ holds by construction, but $|S_1'| + |S_2'| = |S_1| + |S_2| - 1$ due to $v \notin S_1'$, which decreases the number of vertex contacts by one and completes the proof. □

## References


Bourjolly, J.-M., G. Laporte, G. Pesant. 2002. An exact algorithm for the maximum $k$-club problem in an undirected graph. *European Journal of Operational Research* **138**(1) 21–28.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, C. Stein. 2009. *Introduction to Algorithms*. Third edition. MIT Press, Cambridge, MA and London.

Gschwind, T., S. Irnich, I. Podlinski. 2015. Maximum weight relaxed cliques and Russian doll search revisited. Tech. Rep. LM-2015-02, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Held, S., W. Cook, E. C. Sewell. 2012. Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation* **4**(4) 363–381.

Mahdavi Pajouh, F., B. Balasundaram. 2012. On inclusionwise maximal and maximum cardinality $k$-clubs in graphs. *Discrete Optimization* **9**(2) 84–97.

Pattillo, J., N. Youssef, S. Butenko. 2013b. On clique relaxation models in network analysis. *European Journal of Operational Research* **226**(1) 9–18.

Shahinpour, S., S. Butenko. 2013. Algorithms for the maximum $k$-club problem in graphs. *Journal of Combinatorial Optimization* **26**(3) 520–554.

Trukhanov, S., C. Balasubramaniam, B. Balasundaram, S. Butenko. 2013. Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations. *Computational Optimization and Applications* **56**(1) 113–130.

Veremyev, A., V. Boginski. 2012. Identifying large robust network clusters via new compact formulations of maximum $k$-club problems. *European Journal of Operational Research* **218**(2) 316–326.

Wotzlaw, A. 2014. On solving the maximum $k$-club problem. Technical Report arXiv:1403.5111v2, Institut für Informatik, Universität zu Köln, Köln, Germany.