BRANCH-CUT-AND-PROPAGATE FOR THE MAXIMUM k-COLORABLE SUBGRAPH PROBLEM WITH SYMMETRY

TIM JANUSCHOWSKI AND MARC E. PFETSCH

Abstract. Given an undirected graph and a positive integer k, the maximum k-colorable subgraph problem consists of selecting a k-colorable induced subgraph of maximum cardinality. The natural integer programming formulation for this problem exhibits two kinds of symmetry: arbitrarily permuting the color classes and/or applying a non-trivial graph automorphism gives equivalent solutions. It is well known that such symmetries have negative effects on the performance of constraint/integer programming solvers.

We investigate the integration of a branch-and-cut algorithm for solving the maximum k-colorable subgraph problem with constraint propagation techniques to handle the symmetry arising from the graph. The latter symmetry is handled by (non-linear) lexicographic ordering constraints and linearizations thereof. In experiments, we evaluate the influence of several components of our algorithm on the performance, including the different symmetry handling methods. We show that several components are crucial for an efficient algorithm; in particular, the handling of graph symmetries yields a significant performance speed-up.

1. Introduction

Symmetry in integer programs (IPs) and constraint programs (CPs) has been recognized to harm the performance of solution algorithms for a long time. One reason for this unfavorable effect is that symmetric solutions appear repeatedly in the search tree, without giving new information about optimal solutions. Furthermore, in the IP-setting, symmetries usually lead to weak bounds of the linear programming relaxation. In recent years, several (successful) methods to handle symmetry have been developed, see Margot [30] for an overview.

A particular problem in which symmetry arises is the maximum k-colorable subgraph problem, which is defined as follows: Given an undirected graph and a positive integer k, find a largest (induced) subgraph that can be colored with k colors, i.e., we can assign one of at most k colors to each node in the subgraph, such that two adjacent nodes do not receive the same color. Any k-colored subgraph can be transformed into another, equivalent k-colored subgraph by arbitrarily permuting the color classes or by applying a graph automorphism. This gives rise to color and graph symmetry in a natural IP-formulation of the maximum k-colorable subgraph problem.

Date: Jan 2011.

Tim Januschowski is supported by an Embark Scholarship of the Irish Research Council for Science, Engineering and Technology.

One way to tackle symmetry is a polyhedral approach. For instance, so-called *orbitopes* can be used to handle the symmetry arising from permuting the color classes. These polytopes where introduced in [21]. In particular, a complete description via *shifted column inequalities* (SCIs) was given, and an efficient separation algorithm for SCIs was developed. A fast algorithm to perform constraint propagation on orbitope structures was investigated in [20]. We will skip details on these techniques, because they are not of central importance for this paper. They will, however, be used in the computational experiments and are crucial for fast algorithms.

The symmetry arising from automorphisms of the graph, however, has not yet been treated from a polyhedral viewpoint. In this paper, we investigate methods based both on constraint propagation and linear inequalities to handle such symmetry, together with the orbitope approach for color symmetries. To the best of our knowledge, no previous work on combining polyhedral symmetry handling techniques from IP with CP symmetry handling approaches has appeared in the literature. We use the maximum k-colorable subgraph problem as a prototype application for this investigation. This line of research has been started in [18], where we investigate the interaction of the problem-specific polyhedral structure of the maximum k-colorable subgraph problem with orbitopes.

After discussing related work and giving basic definitions, we investigate a symmetry handling approach based on a lexicographic ordering constraint in Section 2.1. For special structures of the automorphism group, one can strengthen the original constraint, for instance, if a transposition is part of the automorphism or a symmetric group is acting on a subgraph. In Section 2.2, we study domain propagation of so-called structural symmetry breaking constraints to handle symmetries that arise from a combination of graph and color symmetries. Section 3 describes components of a branch-cut-and-propagate algorithm to solve the maximum k-colorable subgraph problem, i.e., we combine a branch-and-cut with a constraint programming algorithm to solve a constraint integer program, see [4, 6]. In Section 4, we report on computational experiments, which show the effectiveness of this approach. For example, we show that the usage of symmetric subgroups of the automorphism group lead to a significant improvement of the algorithm performance.

1.1. Related Work

Margot [30] gives an excellent overview of symmetry handling methods in IP and CP. We thus keep this section brief and only highlight some related work.

The handling of symmetries has been extensively discussed in the CP literature. Puget [37] first introduced a symmetry breaking constraint, and Crawford et al. [9] propose general symmetry breaking constraints. The addition of constraints for the handling of row and column symmetry has received considerable attention, see, e.g., [10, 13, 22]. Gent et al. [16] present an overview.

In integer programming, the handling of symmetries by adding constraints was proposed by Méndez-Díaz and Zabala [32] for the graph coloring problem. General IP-methods to handle symmetries via pruning in the search tree have been developed by Margot [27, 28, 29] and Ostrowski et al. [36, 25].

The maximum k-colorable subgraph problem appears to have rarely been studied in the literature, supposedly, because it is closely connected to both the more prominent graph coloring problem and the stable set problem. See [18] for more details as well as a list of applications and relevant IP-techniques for the maximum k-colorable subgraph problem.

1.2. Basic Notation and Definitions

We use the following notation throughout the article.

For an integer $n \geq 1$, let $[n] \coloneqq \{1, 2, \ldots, n\}$. For $x \in \mathbb{R}^{[n]\times[k]}$ and $S \subseteq [n] \times [k]$, we write $x(S) \coloneqq \sum_{(i,j)\in S} x_{ij}$. We use row_i for the set $\{(i,1),(i,2),\ldots,(i,k)\}$ and col_j for the set $\{(1,j),(2,j),\ldots,(n,j)\}$. By $\mathrm{row}_i(x)$ we denote the ith row of x and $\mathrm{col}_j(x)$ denotes the jth column of x. By $\mathbf{0}$ we denote a zero-matrix of appropriate dimensions.

Any permutation of a finite set can be written using disjoint cycles. A *cycle* $(a_1 a_2 \ldots a_k)$ refers to the mapping $a_1 \mapsto a_2 \mapsto \ldots \mapsto a_k \mapsto a_1$; k is the *length* of the cycle, and a cycle of length k is called a *k-cycle*. A 2-cycle is a *transposition*.

Let $G = (V, E)$ be a simple undirected graph with node set V and edge set E, where $n \coloneqq |V|$. We will often need an order of the nodes in V and thus assume that $V \coloneqq \{1, \ldots, n\}$; in particular, we can directly compare nodes, e.g., $u < v$ for $u, v \in V$. We denote by $G[V']$ the subgraph induced by $V' \subseteq V$. Thus, the set of edges of $G[V']$ is $\{e \in E : e \subseteq V'\}$. For a node v, the *neighborhood* $\Gamma(v)$ of node v contains all nodes adjacent to v, i.e.,

$$\Gamma(v) \coloneqq \{u \in V : \{u, v\} \in E\}.$$

The *closed neighborhood* of v is $\overline{\Gamma}(v) \coloneqq \Gamma(v) \cup \{v\}$. The *degree* of v is defined as $\delta(v) \coloneqq |\Gamma(v)|$. A *clique* is a set of nodes $C \neq \varnothing$, such that for all $u, v \in C$, $u \neq v$, we have $\{u, v\} \in E$. A *stable set* is a set of nodes $S \neq \varnothing$ such that for all $u, v \in S$, we have $\{u, v\} \notin E$.

For a positive integer k, G is *k-colorable* if we can assign to each node in G a color (number) in $[k]$ such that adjacent nodes do not have the same color.

Definition 1. *For a positive integer k and a graph G, the* maximum k-colorable subgraph problem *consists of finding a set $V' \subseteq V$ such that $G[V']$ is k-colorable and V' has maximum cardinality.*

The maximum k-colorable subgraph problem is NP-hard and hard to approximate, see [18] for a detailed discussion.

Note that the restriction to simple graphs for the maximum k-colorable subgraph problem is without loss of generality: parallel edges can be replaced by a single edge, and nodes with loops will never be part of V'.

We consider the following IP-formulation for the maximum k-colorable subgraph problem.

$$
\begin{aligned}
(\mathrm{IP}_k(G)) \quad & \max \sum_{v \in V} \sum_{j \in [k]} x_{vj} && && (1)\\
& x_{uj} + x_{vj} \leq 1 && \forall \{u,v\} \in E,\ j \in [k] && (2)\\
& x(\mathrm{row}_v) \leq 1 && \forall\, v \in V && (3)\\
& x_{vj} \in \{0,1\} && \forall\, v \in V,\ j \in [k]. && (4)
\end{aligned}
$$

Let x be a solution of (2)–(4). Because of the *packing inequalities* (3), $x(\mathrm{row}_v)$ is 1 if and only if node v is in the selected subgraph (i.e., colored). Thus, the objective function represents the cardinality of the selected subgraph.

The *k-colored subgraph polytope* corresponding to the maximum k-colorable subgraph problem is

$$\mathrm{P}_k(G) \coloneqq \mathrm{conv}\{x \in \{0,1\}^{V \times [k]} \ : \ x \text{ satisfies (2) and (3)}\}.$$

A *symmetry* of an IP is a permutation of the variables that maps feasible solutions to feasible solutions with the same objective function value. They are variable solution symmetries in the sense of Cohen et al. [8]. The symmetries of an IP form the *symmetry group*. The symmetry group partitions the feasible solutions into disjoint orbits. We say that the addition of a set of constraints leads to *partial/complete* symmetry handling with respect to the symmetry group, if for every orbit of solutions at least/exactly one solution is preserved.

The maximum k-colorable subgraph problem exhibits two basic types of symmetries: color and graph symmetries. In [18], we concentrated on handling color symmetry polyhedrally via orbitopes. Color symmetries form a symmetric group $\mathfrak{S}_k$ of degree k that operates by permuting the columns of x in $(\mathrm{IP}_k(G))$. In this paper, we concentrate on graph symmetry handling.

2. Graph Symmetry Handling

Each element of the automorphism group $\mathrm{Aut}(G)$ of the graph G yields a symmetry of $(\mathrm{IP}_k(G))$ by permuting the rows of x accordingly. Graph and color symmetries differ in certain aspects as we discuss in the following.

First, unlike color symmetries, graph automorphisms may not yield symmetries in the weighted version of the maximum k-colorable subgraph problem, where each node receives a weight and the goal is to maximize the sum of the weights of the colored nodes. Recall that this article only considers the unweighted version.

Second, the color symmetries of $(\mathrm{IP}_k(G))$ are known (but could also be efficiently be computed, see [7]). In contrast, finding graph automorphisms (and hence graph symmetries) is at least as hard as the graph isomorphism problem, which has an open complexity status (see Garey and Johnson [15] and Johnson [19]). More precisely, the problem of detecting whether a graph admits a non-trivial automorphism is graph-isomorphism complete.

Third, whereas color symmetries always form a symmetric group, any group can be the automorphism group of a graph, see, e.g., Frucht [14].

2.1. Lexicographic Graph Symmetry Handling

Crawford et al. [9] propose constraints to handle general symmetries for Boolean satisfiability problems that we can adapt for our purposes as follows. For every symmetry $\phi \in \mathrm{Aut}(G)$, we have the following lexicographic ordering constraint.

$$[\mathrm{row}_1(x), \ldots, \mathrm{row}_n(x)] \geq_{\mathrm{lex}} [\mathrm{row}_{\phi(1)}(x), \ldots, \mathrm{row}_{\phi(n)}(x)]. \tag{5}$$

If we add (5) for all $\phi \in \mathrm{Aut}(G)$, then the graph symmetries are completely handled. In particular, the addition of (5) comes with the guarantee that from every orbit of solutions, the lexicographically largest solution is preserved, see [9]. Note that in (5), it suffices to consider only nodes v for which $\phi(v) \neq v$.

Remark 1. When combining color and graph symmetry handling, one needs to order the rows and the columns of x in the same lexicographic fashion (i.e., decreasingly or increasingly). Orbitope symmetry handling for the color symmetries uses a lexicographically decreasing order on the columns. Hence, a lexicographically decreasing order on the rows should be used as well. Otherwise, one may lose entire orbits of solutions, see Flener et al. [10].

Constraints (5) are non-linear. A standard linearization is:

$$\sum_{i\in[n]} \sum_{j\in[k]} 2^{(n+1-i)k-j}\, x_{ij} \geq \sum_{i\in[n]} \sum_{j\in[k]} 2^{(n+1-i)k-j}\, x_{\phi(i),j}. \tag{6}$$

Inequalities (6) have coefficients from 1 to 2^{nk-1}. Even for small n and k, this will cause severe numerical difficulties for IP-solvers.

We therefore do not use inequalities (6), but directly perform constraint propagation on (5) as follows (see Frisch et al. [13] for domain filtering for (5) in a pure CP context). Assume that x_{ij} becomes fixed during search for some $i \in [n]$, $j \in [k]$; the implications from fixings of $x_{\phi(i),j}$ are similar. Further, assume that x_{st} and $x_{\phi(s),t}$ are fixed for all $s < i$ and all $t \in [k]$, and that

$$[\mathrm{row}_1(x), \ldots, \mathrm{row}_{i-1}(x)] = [\mathrm{row}_{\phi(1)}(x), \ldots, \mathrm{row}_{\phi(i-1)}(x)].$$

If x_{ij} is fixed to 1, this implies that $x_{\phi(i),\ell} = 0$ for $\ell < j$ (otherwise Constraint (5) would be violated). If x_{it} is fixed to 0 for all $t \in [j]$, then we can fix $x_{\phi(i),t} = 0$ for all $t \in [j]$.

To provide further propagation steps, we need the following notation. We denote by $\check{x}$ and $\hat{x}$ the matrix where all unfixed variables in x are set to 0 and 1, respectively. If $\mathrm{row}_{i+1}(\hat{x}) <_{\mathrm{lex}} \mathrm{row}_{\phi(i+1)}(\check{x})$, we can derive stronger fixings. It follows that $\mathrm{row}_i(\tilde{x}) >_{\mathrm{lex}} \mathrm{row}_{\phi(i)}(\tilde{x})$ must hold for every feasible solution $\tilde{x}$ compatible with the fixings, otherwise Constraint (5) is violated. If x_{ij} is fixed to 1, then we may, additionally to the above fixings, fix $x_{\phi(i),j}$ to 0. If x_{it} is fixed to 0 for all $t \in [j]$, then we may, additionally to the above fixings, fix $x_{\phi(i),j+1}$ to 0.

If the automorphism group has a particular structure, this general approach can be specialized. We discuss several examples in the following.

Graph Transpositions: One particularly simple kind of graph automorphisms are transpositions, which frequently occur in the automorphism

group of a graph G. For a transposition $\phi = (u\ v) \in \mathrm{Aut}(G)$ (we always assume $u < v$), we can simplify (5) to:

$$\mathrm{row}_u(x) \geq_{\mathrm{lex}} \mathrm{row}_v(x), \tag{7}$$

and we can simplify (6) to:

$$\sum_{j\in[k]} 2^{k-j} x_{uj} \geq \sum_{j\in[k]} 2^{k-j} x_{vj}. \tag{8}$$

This inequality can be strengthened by using the packing inequalities:

$$\sum_{\ell\in[j]} x_{u\ell} \geq \sum_{\ell\in[j]} x_{v\ell}, \qquad \forall\, j \in [k]. \tag{9}$$

Constraints (8) and (9) are logically equivalent, because they exclude all but the lexicographically largest solutions with respect to $(u\ v)$. However, polyhedrally speaking, (9) dominate (8); they also avoid large coefficients.

Composition of Transpositions: Consider the case in which $\mathrm{Aut}(G)$ contains a composition of (disjoint) transpositions

$$\phi = (u_1\ u_2)(u_3\ u_4)\dots(u_{\ell-1}\ u_\ell),$$

such that $u_i \neq u_{i+1}$ for all $i \in [\ell - 1]$.

In this case, we can strengthen the above mentioned general domain propagation follows. We can assume w.l.o.g. that $u_1 = \min\{u_1, u_2, \dots, u_\ell\}$. In case $\{u_1, u_2\} \in E$, then $x_{u_2,1} = 0$ must hold. For the sake of contradiction, assume that there exists a feasible solution $\tilde{x}$ with $\tilde{x}_{u_2,1} = 1$. It follows that $\tilde{x}_{u_1,1} = 0$ due to $\{u_1, u_2\} \in E$. Then, however, $\tilde{x}$ violates (5); this proves the claim.

Symmetric Groups: In order to handle symmetric (sub)groups, we need the following basic fact.

Lemma 2. *Let $(u\ v) \in \mathrm{Aut}(G)$ with $\{u, v\} \in E$. Then, for any $(v\ w) \in \mathrm{Aut}(G)$, we have $\{v, w\} \in E$.*

Proof. By definition, $\Gamma(u) = \Gamma(v)$ for $(u\ v) \in \mathrm{Aut}(G)$. Because $(u\ w) = (u\ v)(v\ w)(u\ v)$, it follows that $\Gamma(w) = \Gamma(u)$. Since $v \in \Gamma(u) = \Gamma(w)$, we have $\{v, w\} \in E$. □

It follows that a symmetric subgroup of the automorphism group either acts on a stable set or on a clique. If it acts on a clique, we can use orbitopal fixing, see [20], as an efficient domain filtering algorithm; in particular, for a symmetric group acting on the clique $C = \{u_1, u_2, u_3, \dots, u_c\}$, we may fix $x_{u_i,j} = 0$ for all $i = 2, \dots, c$ and $j = 1, \dots, k(i-1)$ due to the lexicographic ordering.

Lemma 3. *Let a symmetric subgroup act on a stable set S. Then adding*

$$x_{uj} = x_{vj} \quad \text{for all } j \in [k],\ u, v \in S, \tag{10}$$

to $(\mathrm{IP}_k(G))$ does not change the optimal value.

Proof. If no optimal solution colors a node in S, the claim holds trivially. Otherwise, assume that there exists a solution $\tilde{x}$ with $\tilde{x}_{uj} = 1$ for some $u \in S$ and $j \in [k]$. Consider any node $v \in S \setminus \{u\}$. By assumption, we have $(u\ v) \in \mathrm{Aut}(G)$, i.e., $\Gamma(u) = \Gamma(v)$. Since $\tilde{x}$ is a valid coloring, there exists

no $w \in \Gamma(v) = \Gamma(u)$ such that $\tilde{x}_{wj} = 1$. Since by assumption $\{u, v\} \notin E$, (re)coloring v with color j yields a valid coloring. Since v was arbitrary, we conclude that any node in S can be colored with the same color. Moreover, if $\tilde{x}$ was optimal the resulting coloring is optimal and satisfies (10). □

We note that the equations (10) do not conflict with Constraints (5) for optimal solutions: Every solution that fulfills (10) has lexicographically ordered rows for nodes in S and thus fulfills (5). Moreover, the proof of Lemma 3 shows that if a solution fulfills (5) but not (10), the solution is either not optimal or can be partially recolored such that it fulfills (10).

Arbitrary Groups: In the general case, we cannot exploit any particular structure of the automorphism group. In particular, since $\mathrm{Aut}(G)$ may become very large, it is in general not efficient to consider (5) for all available automorphisms. Thus, we only consider the generators of $\mathrm{Aut}(G)$ that NAUTY [31] outputs.

2.2. Combinations of Graph and Color Symmetry

So far, we have focused on the independent handling of graph and color symmetries: orbitopes handle the color symmetries completely, and (5) can handle graph symmetries completely. As the following example shows, there may, however, be combinations of graph and color symmetries (*product symmetries*) that are left unhandled, even if we restrict attention to transpositions (see also Flener et al. [10]). In this section we deal with methods to handle product symmetries.

Example 4. Consider a graph with three isolated nodes $\{1, 2, 3\}$, $k = 3$, and the two solutions $[1, 0, 0; 0, 1, 0; 0, 1, 0]$ and $[1, 0, 0; 1, 0, 0; 0, 1, 0]$ (with the obvious interpretation as matrices). Both solutions have lexicographically ordered rows and columns, but are symmetric via a combination of graph and color symmetries.

Note that the difference between the number of solutions in the case with and without complete handling of product symmetries can be exponential in the size of G, even if graph and color symmetries are handled completely, see [22].

Flener et al. [11] present constraints that provide complete symmetry handling for product symmetries, where the graph automorphism group is a union of symmetric groups. We need some notation in order to present this result. We partition the set of nodes V into sets $V_1, \dots, V_s$ such that a symmetric subgroup $\mathfrak{S}_p$ of $\mathrm{Aut}(G)$ acts on V_p for all $p \in [s]$. Let $V_p = \{v_1^p, \dots, v_{q_p}^p\}$ with $v_1^p < v_2^p < \cdots < v_{q_p}^p$. We require the lexicographic ordering

$$\mathrm{row}_{v_i^p}(x) \geq_{\mathrm{lex}} \mathrm{row}_{v_{i+1}^p}(x), \tag{11}$$

for all $i \in [q_p - 1]$ and all $p \in [s]$. We introduce *frequency variables*

$$f_j^p \coloneqq \sum_{v \in V_p} x_{vj} \in \mathbb{Z}_+, \ j \in [k], \ p \in [s], \tag{12}$$

determining the number of nodes in V_p colored with color j. Finally, we introduce the constraints

$$(f_j^1, f_j^2, f_j^3, \dots, f_j^s) \geq_{\text{lex}} (f_{j+1}^1, f_{j+1}^2, f_{j+1}^3, \dots, f_{j+1}^s),\ j \in [k-1]. \tag{13}$$

We refer to Constraints (11) and (13) as *structural symmetry breaking* (SSB) constraints. Note that variables f_j^p are only used for ease of exposition; we do not use them in our implementation. Flener et al. [11] proved that SSB constraints completely handle all symmetries of $(\text{IP}_k(G))$.

Unfortunately, SSB constraints do not necessarily imply a lexicographic ordering of the columns as one easily verifies. This impedes a combination of orbitope symmetry handling with SSB constraints in the general case. However, the following result shows how to relabel the nodes such that orbitope symmetry handling and SSB constraints do not conflict.

Proposition 5. *Let G be a graph such that a symmetric subgroup $\mathfrak{S}_p$ of $\text{Aut}(G)$ acts on V_p for $p = 1, \dots, s$. Let the node labeling of G be such that*

$$v_1^1 < \cdots < v_{q_1}^1 < v_1^2 < \cdots < v_{q_2}^2 < v_1^3 < \cdots < v_1^s < \cdots < v_{q_s}^s.$$

Then SSB constraints imply a lexicographically decreasing order on the columns.

Proof. For the sake of contradiction, assume the existence of a solution $(\tilde{x}, \tilde{f})$ that fulfills (13) and has two columns j, $j+1$ such that $\text{col}_j(\tilde{x}) <_{\text{lex}} \text{col}_{j+1}(\tilde{x})$. Let i be the smallest row index in which $\text{col}_j(\tilde{x})$ and $\text{col}_{j+1}(\tilde{x})$ differ and $\tilde{x}_{ij} = 0$, $\tilde{x}_{i,j+1} = 1$. Let node i be in partition V_p.

By assumption, $\tilde{x}_{rj} = \tilde{x}_{r,j+1}$ for $r < i$. Because of the packing inequalities, it follows that $\tilde{x}_{rj} = \tilde{x}_{r,j+1} = 0$. Thus, $\tilde{f}_j^q = \tilde{f}_{j+1}^q = 0$ for all $q \in [p-1]$, due to the node labeling.

Within partition V_p, we have $\tilde{x}_{rj} = 0$ for all $r \in V_p$ with $r < i$. Moreover, node i is colored with color $j+1$. Thus, $\tilde{x}_{it} = 0$ for all $t \in [j]$ by the packing inequalities. Since the rows within a partition are lexicographically decreasingly ordered, it follows that $\tilde{x}_{rt} = 0$ for all r in V_p with $r > i$ and all $t \in [j]$. Therefore, we have shown that $\tilde{x}_{vj} = 0$ for all $v \in V_p$. It follows that $0 = \tilde{f}_j^p < \tilde{f}_{j+1}^p \geq 1$. Thus, $\tilde{x}$ violates an SSB constraint. □

The node labeling/order is important for the maximum k-colorable subgraph problem and has a significant influence on the solving time. We shall verify in experiments, whether the additional symmetry handling by SSB constraints outweighs the effect of restricted node orderings.

We directly perform domain filtering for SSB constraints as follows (see Flener et al. [11] for stronger domain filtering in a pure CP context). First, for Constraint (11), we use the already mentioned domain propagation of Constraint (5). Second, we propagate Constraint (13) as follows.

Assume variable x_{ij} becomes fixed, with corresponding frequency variable f_j^p. Assume further that all variables x_{vj} and $x_{v,j+1}$ for all $v \in V_q$ are fixed such that $f_j^q = f_{j+1}^q$ for all $q < p$.

First, consider the case where all variables $x_{v,j+1}$ with $v \in V_p$ are fixed. Then $x(V_p \times \{j+1\}) = f_{j+1}^p$, and because of (13), it follows that $\tilde{x}(V_p \times \{j\}) \geq f_{j+1}^p$ must hold for any feasible $\tilde{x}$ that is compatible with the fixings. Assume that $\hat{x}(V_p \times \{j\}) = \ell < f_{j+1}^p$ (see Section 2.1 for a definition of $\hat{x}$). If there

are $(f^p_{j+1} - \ell)$ unfixed variables with indices in $V_p \times \{j\}$, we can fix all of them to 1. If there are strictly fewer, the constraint is violated; if there are more, we do nothing.

Moreover, consider the case where $\hat{x}(V_{p+1} \times \{j\}) < \check{x}(V_{p+1} \times \{j+1\})$, then $f^p_j > f^p_{j+1}$ must hold for all solutions compatible with the fixings. If there are $(f^p_{j+1} - \ell + 1)$ unfixed variables with indices in $V_p \times \{j\}$, we can fix all of them to 1. If there are strictly fewer, the constraint is violated; if there are more, we do nothing. We can deduce similar domain filterings for variables in $V_p \times \{j-1\}$ based on the fixing of x_{ij}.

3. A Branch-Cut-and-Propagate Algorithm

In this section, we describe the main components of a branch-cut-and-propagate algorithm for the maximum k-colorable subgraph problem: preprocessing, node labeling heuristics, branching rules, (valid) inequalities, and node domination inequalities. Additionally, we use the constraints for graph symmetries as described in Section 2 and conflict analysis, see Achterberg [3, 4]. We assume familiarity with the branch-and-cut approach, see, e.g., Nemhauser and Wolsey [35] for background.

At several places, we use TCLIQUE, a specialized branch-and-bound method for the maximum weighted clique problem, which is available in SCIP [38]. By working on the complement graph, TCLIQUE can solve the maximum weighted stable set problem as well.

3.1. Preprocessing

Preprocessing tries to reduce the size of the graph, before the main optimization is performed. For the maximum k-colorable subgraph problem, we can remove nodes with low degree as follows. If a node v with $\delta(v) < k$ exists, we can always color this node with a color that is not used by its neighbors. Hence, we can delete this node from graph G and adapt the objective function.

3.2. Node Labeling Heuristics

Before setting up formulation ($\mathrm{IP}_k(G)$), one can reorder (relabel) the nodes, which has a big impact on the performance of solution algorithms. We investigate the following node labelings:

- Sort nodes w.r.t. increasing/decreasing node-degree.
- Choose a large clique as the first nodes in the order and then sort w.r.t. increasing/decreasing node-degree.
- Sort the nodes such that nodes in the same symmetric group have consecutive node labels (see Proposition 5).
- Sort the nodes randomly.

3.3. Branching Rules

One possibility is to branch on nodes as done in [33], i.e., we choose a node in the graph and generate a branch for each color which is still available.

- *Uncolored node branching*: choose a node with highest number of non-available colors. Ties are broken according to the degree of the nodes in the uncolored subgraph, i.e., the number of neighbors whose color is not fixed.
- *Uncolored Sewell's rule*: choose a node as above. Ties are broken according to the rule of Sewell [39]: The node whose coloring induces the smallest number of colorings for its neighborhood is selected.

As an alternative, we branch on the first fractional *variable* in the row-wise ordering (*first index branching*), see [20]. In the *truncated first index branching* variant, we branch on the first fractional variable x_{ij} for which the current LP-solution has no 1 in positions $(j,j),\dots,(i-1,j)$. If no such variable has been found, we use the standard (strong-)branching rule of SCIP, see [5]. These two variable branching rules try to support the lexicographic ordering of the columns.

3.4. Valid Inequalities

In order to generate additional cutting planes, we implemented the following separation algorithms.

- Let S be a set of nodes in G, and let $\alpha(S)$ be the maximum size of a stable set in $G[S]$. Then, the inequality
$$\sum_{v\in S} x_{vj} \le \alpha(S) \tag{14}$$
is a valid inequality for $\mathrm{P}_k(G)$ for all $j \in [k]$. We use a heuristic algorithm to separate (14) with small S. If S is a clique, (14) are called *clique inequalities*; we use TCLIQUE to separate clique inequalities. If C is an odd cycle in G, then we refer to (14) as an *odd cycle inequality*. We separate odd cycle inequalities with an algorithm due to Grötschel et al. [17].
- The *neighborhood inequality* for a node v is given by
$$\sum_{u\in\Gamma(v)} x_{uj} + r x_{vj} \le r \qquad \forall\, j \in [k],$$
where $r = \alpha(\Gamma(v))$, see [33, 34]; r is computed via TCLIQUE. These inequalities are added to $\mathrm{IP}_k(G)$ from the start.
- *Shifted Column Inequalities* (SCIs) for orbitopes are separated with the linear-time separation algorithm described in [21].
- *Clique Shifted Column Inequalities*, see [18, 21], and *Packing-Clique Inequalities*, see [18], are separated heuristically.

3.5. Dominated Nodes

Additional inequalities can be added at the start by considering dominated nodes. A node v *dominates* another node u if $\overline{\Gamma}(v) \supseteq \Gamma(u)$. We say that v *strictly* dominates u, if $\Gamma(v)\setminus\{u\} \supsetneq \Gamma(u)\setminus\{v\}$. Dominated nodes appear in the context of the stable set problem, see, e.g., [12], or the graph coloring problem, see, e.g., [26]. In fact, in graph coloring, we can remove dominated nodes from the graph if $\{u,v\} \notin E$, because they can always be colored with the same color as the dominating nodes. In the maximum k-colorable

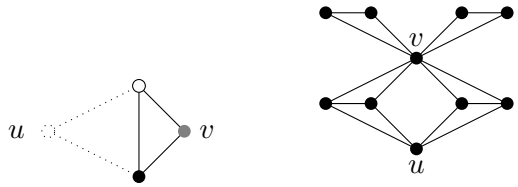

Figure 1: Node v dominates node u. *Left:* For $k = 2$, removing node u from the graph could lead to a solution that is not easily extensible (black and white node) to the original graph. *Right:* Coloring the dominated node u leads to a non-optimal solution.

subgraph problem one can show that this is not the case, as the following example shows.

Example 6. In this example, we show that removing dominated nodes from the graph and then solving the remaining problem to optimality will in general not lead to an optimal solution of the whole problem (see Section 3.5).

We consider the graphs depicted in Figure 1 and let $k = 2$. In both graphs node v dominates node u. On the left-hand side of Figure 1, if we remove node u, we may obtain an optimal solution that consists of the two neighbors of u. However, an optimal solution of the whole graph contains nodes u, v and one of the neighbors that nodes u and v have in common. This example shows that it is not straight-forward to extend a solution of a graph with removed dominated nodes to the original graph. Next, we show that there exists a graph such that no optimal solution contains a dominated node. In the graph on the right-hand side of Figure 1, for $k = 2$, any solution that colors node u can contain at most 7 nodes, however, a solution with 8 nodes exists.

We can, however, generate inequalities based on the following result.

Theorem 7. *Let $u, v \in V$, $u \neq v$, be a pair of nodes with $\overline{\Gamma}(v) \supseteq \Gamma(u)$.*

(1) *If $\{u, v\} \in E$, then adding the following inequality does not change the optimal value of* $(\mathrm{IP}_k(G))$*:*

$$\sum_{j \in [k]} x_{vj} \leq \sum_{j \in [k]} x_{uj}. \tag{15}$$

(2) *If $\{u, v\} \notin E$, then adding the following inequalities does not change the optimal value of* $(\mathrm{IP}_k(G))$*:*

$$x_{vj} \leq x_{uj} \quad \forall\, j \in [k]. \tag{16}$$

To provide a proof of Theorem 7, which deals with inequalities arising from dominated nodes, we first prove the following result.

Proposition 8. *Let $u, v \in V$, $u \neq v$, be a pair of nodes with $\overline{\Gamma}(v) \supseteq \Gamma(u)$.*

(1) *If $\{u, v\} \in E$, then for any $x \in \mathrm{P}_k(G)$ with $x(\mathrm{row}_v) = 1$, there exists a solution $x' \in \mathrm{P}_k(G)$ with $x'(\mathrm{row}_u) = 1$ and $\mathbb{1}^T x = \mathbb{1}^T x'$.*

(2) *If $\{u, v\} \notin E$, then for any $x \in \mathrm{P}_k(G)$ with $x(\mathrm{row}_v) = 1$, there exists a solution $x' \in \mathrm{P}_k(G)$ with $x'(\mathrm{row}_u) = x'(\mathrm{row}_v) = 1$. For any optimal solution $x \in \mathrm{P}_k(G)$ with $x(\mathrm{row}_v) = 1$, there exists an optimal solution $x' \in \mathrm{P}_k(G)$ with $x'_{uj} = x'_{vj}$ for all $j \in [k]$.*

Figure 2: *Left:* All pairs of nodes u, v dominate each other, i.e., $\overline{\Gamma}(v) \supseteq \Gamma(u)$. *Right:* Node u is dominated by nodes v and w. For $k = 2$, an optimal 2-colored subgraph is depicted in gray and black. It contains u, v, and w. However, we cannot color node u with the same color as v *and* w.

Proof. (1) Assume that a solution x of the maximum k-colorable subgraph problem exists that contains node v, but not node u, otherwise there is nothing to prove. By assumption, $\overline{\Gamma}(v) \supseteq \Gamma(u)$, so if we delete node v from the solution, we can always color node u with the former color of node v and obtain a solution with the same number of nodes.

(2) Assume that a solution x of the maximum k-colorable subgraph problem exists that contains node v, but not node u. Now color node u with the same color as node v, say color j. We claim that this is a valid coloring: Since $\Gamma(v) \supseteq \Gamma(u)$, node u is not adjacent to any node colored with color j; in particular, since $\{u, v\} \notin E$, we have $v \notin \Gamma(u)$. Using this argument, for an optimal solution that contains node v colored with color j and node u with color $t \neq j$, we can simply color node u with color j to obtain a valid, equivalent solution. □

Proof of Theorem 7. (1) Part 1 of Proposition 8 shows that if there exists an optimal solution that colors v, there exists an optimal solution that colors u. Inequality (15) picks the latter one, if it exists.

(2) Inequalities (16) express the fact that if there exists an optimal solution coloring v, there exists an optimal solution in which u has the same color as v. This is guaranteed by Part 2 of Proposition 8. □

If node u dominates v and v dominates u, this induces a graph transposition $(u\ v) \in \mathrm{Aut}(G)$. For graph transpositions, we have already provided stronger inequalities: Inequalities (9) for $\{u, v\} \in E$ and Equation (10) for $\{u, v\} \notin E$. In particular, we can add (9) and (10) for *all* such pairs u, v with $u < v$.

Avoiding Conflicts between Constraints: Adding Inequalities (15) for all pairs of (strictly) dominated/dominating nodes may cut off all optimal solutions. The left-hand side of Figure 2 shows an example: If $k = 2$, including (15) for all pairs cuts off all optimal solutions. Similarly, we cannot add all Inequalities (16), see the right-hand side of Figure 2.

In order to avoid these difficulties, we construct a directed graph D. For every pair of nodes u, v, such that v strictly dominates u, we add a directed edge (v, u). We obtain the following result.

Lemma 9. *Let A be a subset of arcs of D such that A is a branching, i.e., A is acyclic and the in-degree of every node is at most one. Then at least one optimal solution remains when adding* (15) *and* (16) *for all arcs in A.*

Table 1: Partial list of benchmark instances. On the left part are Color02 instances; on the top right DIMACS clique instances, and on the bottom right complement graphs of DIMACS clique instances.

Name	k_1	k_2	k_3	k_4
1-FullIns_4	3			
1-Insertions_4	3			
2-FullIns_4	3			
3-FullIns_4	5	6		
4-FullIns_3	3			
4-FullIns_4	5	6	7	
5-FullIns_3	3			
5-FullIns_4	5	6	7	8
DSJC125.9	4	5	6	
DSJC250.9	3	4		
DSJR500.1c	3	4	5	
DSJR500.1	8	9	10	11
myciel5	4	5		
myciel6	3			
queen6_6	6			

Name	k_1	k_2	k_3	k_4	k_5	k_6
san200_0.9_1	4					
c-fat500-2	15	17	20	22	25	
c-fat500-10	10	15	17	20	22	25
gen200_p0.9_55	4					
c-fat200-1	10					
gen200_p0.9_44	4					
sanr200_0.9	4					
san200_0.9_2	4					
MANN_a27	15	17	20	22	25	
johnson8-4-4c	4					
c-fat500-1c	4	5	6	6		
c-fat200-2c	7	8				
c-fat200-1c	6	7	8	9		
hamming6-4c	4	5				

Proof. Consider an optimal solution $\tilde{x}$ that violates (15) or (16) for a pair of nodes u, v, such that v dominates u and $(v,u) \in A$. Since A forms a branching, there exists a unique root r in the connected component C of A that contains u and v. W.l.o.g. assume that (v,u) is an arc with smallest distance to r in C, among all arcs in A such that $\tilde{x}$ violates the corresponding inequalities (15) or (16).

Following the proof of Proposition 8, we will change $\tilde{x}$ by (re)coloring u in such a way that a new arc $(z,w) \in A$ for which inequality (15) or (16) is violated has strictly larger distance from r than (v,u) and the inequalities for (v,u) are satisfied.

Consider first the case where $\{u,v\} \in E$, and assume that v receives color j (if v is uncolored, Inequality (15) is not violated). We (re)color u with color j and uncolor v. This yields a feasible (optimal) coloring, see the proof of Proposition 8. Uncoloring v does not invalidate (15) or (16). (Re)coloring u might violated some inequality for a pair $(u,w) \in A$. This arc has strictly larger distance to r than arc (v,u).

Similarly, if $\{u,v\} \notin E$ we (re)color node u with color j. This yields a feasible (optimal) coloring, see again the proof of Proposition 8. As above, (re)coloring u might violate some inequality for a pair $(u,w) \in A$. This arc has strictly larger distance to r than (v,u).

We conclude that repeating this process must eventually lead to an optimal solution that does not violate any inequalities (15) or (16). □

Similarly, a naïve combination of constraints for graph transposition or dominated neighborhood inequalities may remove all optimal solutions in ($\mathrm{IP}_k(G)$). Using an analogous construction as above guarantees that at least one optimal solution remains.

4. Computational Results

We implemented a branch-cut-and-propagate algorithm as described above in C++ based on SCIP 2.0.0, see [4, 38]. We use CPLEX 12.1 as the LP-solver. The experiments were run on a linux cluster, in which each node has 8 Intel Xeon CPUs with 2.66GHz; we only use four CPUs per node to avoid timing issues.

To obtain a test set, we selected graphs from the Color02 symposium [2] and the clique part of the Second DIMACS Implementation Challenge [1] benchmarks; we add the complement of some of the clique graphs. Finally, we included instances for the wave length assignment problem, see [23, 24]; only instances for which the chromatic number is strictly greater than 20 were considered for $k = 20$. In total, we selected 74 combinations of graphs and numbers of colors; see Table 1 for the Color02 and DIMACS graphs (for more details see Table 2). The instances were chosen such that our default version (described below) can prove optimality of each instance within about one hour.

Table 2 gives more details on the instances that we use to evaluate the performance of our branch-cut-and-propagate algorithm. Column 'n' denotes the number of nodes of the instance and 'm' the number of edges. We also report the approximate size of the graph automorphism group in column '$|\mathrm{Aut}(G)|$', the number of generators in the group in '$\#g$' and the number of transpositions in '$\#t$'; all of these numbers were found with NAUTY [31]. Finally, by (k_i, α_i), we denote the number of colors k_i together with the size of the optimal solution α_i.

In our experiments, we use a time-limit of two hours. We initialize the algorithm with an optimal solution in order to minimize the effect of heuristics. Table 3 sums up the experiments. We report the number of instances that could not be solved within two hours (column '>2h'), and we report the shifted geometric mean[1] of the number of nodes in the search tree as well as the shifted geometric mean of the CPU time in seconds (columns 'nodes' and 'time', respectively). If an instance times out, its running time is evaluated as 2h, and we use the number of nodes produced within 2h for the computation of the geometric means.

Our default settings are as follows:

Preprocessing: Remove nodes with low degree smaller than k.
Node labeling: Choose a clique as the first nodes in the labeling, then label nodes with decreasing degree.
Cutting Planes: Neighborhood inequalities are added to $\mathrm{IP}_k(G)$, stable set inequalities (14) are separated heuristically in the root node, and clique inequalities are separated at every fifth depth of the tree.
Symmetry Handling: Use orbitopal fixing for color symmetries.
Branching Rule: Use the uncolored Sewell's rule.

We separate cutting planes other than clique inequalities only at the root node; this seems to be generally superior for our test set. In the following

[1]The shifted geometric mean of values $t_1, \ldots, t_n$ is defined as $\left(\prod(t_i + s)\right)^{1/n} - s$ with shift s. We use a shift $s = 10$ for time and $s = 100$ for nodes in order to decrease the strong influence of the very easy instances in the mean values.

Table 2: Benchmark instances. In the first part are Color02 instances, in the second part DIMACS clique instances, in the third part complement graphs of clique instances, and finally wave length assignment problem instances.

Name	n	m	$\lvert\mathrm{Aut}(G)\rvert$	$\#g$	$\#t$	(k_1, α_1)	(k_2, α_2)	(k_3, α_3)
1-FullIns_4	93	593	16	4	1	(3, 87)		
1-Insertions_4	67	232	14	2	0	(3, 63)		
2-FullIns_4	212	1621	16	4	1	(3, 202)		
3-FullIns_4	405	3524	16	4	1	(5, 402)	(6,404)	
4-FullIns_3	114	541	8	3	1	(3, 106)		
4-FullIns_4	690	6650	16	4	1	(5, 684)	(6, 687)	(7, 689)
5-FullIns_3	154	792	8	3	1	(3, 144)		
5-FullIns_4	1085	11395	16	4	1	(5, 1076)	(6, 1079)	
						(7, 1082)	(8, 1084)	
DSJC125.9	125	6961	1	0	0	(4, 16)	(5, 20)	(6, 23)
DSJC250.9	250	27897	1	0	0	(3, 14)	(4, 18)	
DSJR500.1c	500	121275	$>1{\cdot}10^{7}$	23	23	(3, 37)	(4 48)	(5, 59)
DSJR500.1	500	3555	98304	16	16	(8, 459)	(9, 477)	
						(10, 489)	(11, 496)	
myciel5	47	236	10	2	0	(4, 44)	(5,46)	
myciel6	95	755	10	2	0	(3, 83)		
queen6_6	36	290	8	3	0	(6, 32)		
san200_0.9_1	200	17910	1	0	0	(4, 16)		
c-fat500-2	500	9139	$>6{\cdot}10^{369}$	460	459	(15, 300)	(17, 340)	
						(20, 400)	(22, 440)	(25, 490)
c-fat500-10	500	46627	$>3{\cdot}10^{691}$	492	491	(10, 40)	(15, 60)	(17, 68)
						(20, 80)	(22, 88)	(25, 100)
gen200_p0.9_55	200	17910	1	0	0	(4, 17)		
c-fat200-1	200	1534	$>7{\cdot}10^{88}$	163	162	(10, 180)		
gen200_p0.9_44	200	17910	1	0	0	(4, 20)		
sanr200_0.9	200	17863	1	0	0	(4, 16)		
san200_0.9_2	200	17910	1	0	0	(4, 16)		
MANN_a27	378	70551	303264	8	0	(15, 45)	(17, 51)	(20, 60)
						(22, 66)	(25, 75)	
johnson8-4-4c	70	560	80640	5	0	(4, 52)		
c-fat500-1c	500	120291	$>6{\cdot}10^{2454}$	420	419	(4, 56)	(5, 70)	
						(6, 84)	(6, 134)	
c-fat200-2c	200	16665	$>1{\cdot}10^{139}$	182	181	(7, 156)	(8, 178)	
c-fat200-1c	200	18366	$>7{\cdot}10^{88}$	163	162	(6, 72)	(7, 84)	
						(8, 95)	(9, 105)	
hamming6-4c	64	1312	46080	7	0	(4, 16)	(5, 20)	
wa612	187	3340	$>3{\cdot}10^{7}$	23	23	(20, 185)		
wa614	201	3570	$>1{\cdot}10^{10}$	30	30	(20, 199)		
wa620	205	3543	$>1{\cdot}10^{10}$	30	30	(20, 203)		
wa622	212	3505	$>3{\cdot}10^{12}$	37	37	(20, 210)		
wa707	235	4717	$>2{\cdot}10^{11}$	33	33	(20, 234)		
wa712	237	4458	$>2{\cdot}10^{13}$	35	35	(20, 236)		
wa815	275	5646	$>1{\cdot}10^{13}$	42	42	(20, 274)		
wa822	268	5410	$>7{\cdot}10^{11}$	36	36	(20, 267)		
wa915	290	6076	$>7{\cdot}10^{15}$	48	48	(20, 288)		

Table 3: Summary of experiments

Variant	>2h	nodes	time
default	0	251.0	104.7
OFSCI	1	275.4	127.7
noOFSCI	2	329.3	111.2
noOFnoSCI	16	1462.5	449.5
CliqueSCI	9	284.2	238.8
PackingClique	0	275.3	117.7
IncDeg	19	1246.0	480.9
DecDeg	20	1526.7	431.2
CliqueIncDeg	2	378.7	135.7
SymGOrder	14	848.2	312.5
Random	17	1269.7	445.3

Variant	>2h	nodes	time
firstIndex	1	288.5	126.2
TruncFirstIndex	0	192.0	108.8
UCVertex	0	268.1	109.4
OddCycle	0	296.9	145.3
DomNode	1	292.2	112.9
DomNodeG$\mathfrak{S}$	0	245.1	85.7
G$\mathfrak{S}$	0	249.1	84.0
GGen	0	218.0	86.9
G$\mathfrak{S}$SymGOrder	14	832.5	273.8
GGenSymGOrder	13	748.1	267.4
SSB	13	869.9	270.2
GGenSSB	13	808.5	274.6

experiments, we add/remove features to the default settings in order to asses the importance of the features on the overall performance.

Experiment 1: Color Symmetry Handling. In the first experiment, we evaluate the effect of different degrees of color symmetry handling: separating SCIs with/without orbitopal fixing (OFSCI/noOFSCI in Table 3), no color symmetry handling (noOFnoSCI), additionally separating clique shifted column inequalities (CliqueSCI), and the packing-clique inequalities (PackingClique). The default is to not separate SCIs, but perform orbitopal fixing.

Any form of color symmetry handling clearly outperforms the variant without color symmetry handling. The default settings are the fastest. Orbitopal fixing as a pure constraint propagation algorithm outperforms the variants that separate shifted column inequalities (the same is observed in [20]), which can be explained by the fact that separating SCIs seems to yield more difficult LPs. The strengthened inequalities (PackingClique or CliqueSCI) do not yield an improvement over the default or variant 'noOFSCI'.

Experiment 2: Node Labelings. Here we investigate the effect of the node labelings on the performance. Our results clearly indicate their high impact. The default settings are the best choice. The other variants increasing/decreasing degree (IncDeg/DecDeg), keep symmetry groups together (SymGOrder), and random ordering (Random) all perform much worse. Note that sorting a clique to the front, as done in the default settings, has a high positive impact.

Experiment 3: Branching Rules. We compare the two vertex based branching rules 'uncolored vertex' (UCVertex) and 'uncolored Sewell' (default) with the variable branching rules 'first index' (FirstIndex) and 'truncated first index' (TruncFirstIndex). Here, 'first index' has the worst performance, while the other variants are not far apart w.r.t. their running time. The variant 'truncated first index' produces the fewest number of nodes

in the tree, but is not the fastest, probably because of the time used for strong-branching.

Experiment 4: Cutting Planes. In this experiment, we compare separating clique inequalities (default) with the additional separation of odd cycle inequalities at the root node (OddCycle). Clearly, separating odd cycle inequalities does not yield an improvement to the default settings.

Experiment 5: Graph Symmetry Handling. In the final experiment, we compare different forms of graph symmetry handling and dominated node inequalities. We evaluate handling dominated nodes (DomNodes), symmetric subgroups of Aut(G) (G$\mathfrak{S}$), their combination (DomNodeG$\mathfrak{S}$), generator symmetry (GGen) which subsume graph 'G$\mathfrak{S}$', symmetric subgroups with SSB Constraints (SSB), and combining generator symmetry with SSB Constraints (GGenSSB). Whenever we use SSB constraints, we use the node labeling 'SymGOrder' in order to be able to combine SSB Constraints with orbitopal fixing. We further present the results for G$\mathfrak{S}$ and GGen w.r.t. the labeling 'SymGOrder' (G$\mathfrak{S}$SymGOrder/GGenSymGOrder).

Handling dominated nodes deteriorates the performance of the default settings; however, handling symmetric subgroups additionally yields an improvement on the default version. The overall best settings are obtained by handling symmetric subgroups only. The gain in time over the default version is approximately 20% in running time. The strongest form of graph symmetry handling (GGen) produces the least of amount of search nodes for the default branching rule; however, the running time is slightly worse than handling symmetric subgroups only.

Any graph symmetry handling in combination with labeling 'SymGOrder' clearly outperforms the variant with labeling 'SymGOrder' without graph symmetry handling. However, the handling of additional product symmetries via SSB constraints does not seem to have a positive impact on the solving time. The difference in running time is fairly small for all graph symmetry handling with labeling 'SymGOrder'. Overall they are inferior to graph symmetry handling with the default labeling, e.g., version 'G$\mathfrak{S}$' vs. 'G$\mathfrak{S}$SymGOrder'.

5. Conclusion and Future Work

The branch-cut-and-propagate algorithm that we presented in this paper provides an efficient way to solve a symmetric formulation of the maximum k-colorable subgraph problem. The performance of the algorithm can be substantially improved by handling symmetries via a combination of inequalities and domain propagation. In particular, we have demonstrated that handling graph symmetries in this way is very useful.

Future work will involve a more detailed computational analysis of the results in this paper. Moreover, one could identify (strong) linearizations for lexicographic ordering constraints for general graph symmetries and SSB constraints; this would help to polyhedrally handle additional symmetries.

References

[1] Second DIMACS implementation challenge: Maximum clique, graph coloring, and satisfiability. Available at `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique/`, 1993.

[2] COLOR02 – computational symposium: Graph coloring and its generalizations. Available at `http://mat.gsia.cmu.edu/COLOR02`, 2002.

[3] T. Achterberg. Conflict analysis in mixed integer programming. *Discrete Optimization*, 4(1):4–20, 2007.

[4] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1), 2009.

[5] T. Achterberg and T. Berthold. Hybrid branching. In W.-J. van Hoeve and J. Hooker, editors, *CPAIOR 2009*, volume 5547 of *LNCS*, pages 309–311. Springer-Verlag, 2009.

[6] T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In *Proceedings of the Fifth International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR '08)*, volume 5015 of *LNCS*, pages 6–20. Springer, 2008.

[7] T. Berthold and M. E. Pfetsch. Detecting orbitopal symmetries. In *Operations Research Proceedings 2008*, pages 433–438. Springer, 2009.

[8] D. Cohen, P. Jeavons, C. Jefferson, K. Petrie, and B. Smith. Symmetry definitions for constraint satisfaction problems. *Constraints*, 11:115–137, 2006.

[9] J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR '96)*, pages 148–159. Morgan Kaufmann, 1996.

[10] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, and T. Walsh. Breaking row and column symmetries in matrix models. In *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming (CP '02)*, pages 462–476. Springer, 2002.

[11] P. Flener, J. Pearson, and M. Sellmann. Static and dynamic structural symmetry breaking. *Annals of Mathematics and Artificial Intelligence*, 57(1):37–57, 2009.

[12] F. V. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56:25:1–25:32, 2009.

[13] A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Propagation algorithms for lexicographic ordering constraints. *Artificial Intelligence*, 170:803–834, 2006.

[14] R. Frucht. Herstellung von Graphen mit vorgegebener abstrakter Gruppe. *Compositio Mathematica*, 6:239–250, 1938.

[15] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[16] I. P. Gent, K. E. Petrie, and J.-F. Puget. Symmetry in constraint programming. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, pages 329–376. Elsevier, 2006.

[17] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, Heidelberg, 2nd edition, 1993.

[18] T. Januschowski and M. E. Pfetsch. The maximal k-colorable subgraph problem and orbitopes. `http://www.optimization-online.org/DB_HTML/2010/11/2821.html`, 2010.

[19] D. S. Johnson. The NP-completeness column: An ongoing guide. V. *Journal of Algorithms*, 3:381–395, 1982.

[20] V. Kaibel, M. Peinhardt, and M. E. Pfetsch. Orbitopal fixing. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization, Proceedings of the Twelfth International IPCO Conference*, volume 4513 of *LNCS*, pages 74–88. Springer, 2007.

[21] V. Kaibel and M. E. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

[22] G. Katsirelos, N. Narodytska, and T. Walsh. On the complexity and completeness of static constraints for breaking row and column symmetry. In D. Cohen, editor, *Proceedings of the Eighteenth International Conference on Principles and Practice of Constraint Programming (CP '10)*, pages 305–320. Springer, 2010.

[23] A. M. C. A. Koster and S. Ruepp. Benchmarking RWA strategies for dynamically controlled optical networks. In *Proceedings of the Thirteenth International Telecommunications Network Strategy and Planning Symposium (NETWORKS '08)*, pages 1–14, 2008.

[24] A. M. C. A. Koster and M. Scheffel. A routing and network dimensioning strategy to reduce wavelength continuity conflicts in all-optical networks. In *Proceedings of the International Network Optimization Conference (INOC '07)*, Spa, Belgium, 2007.

[25] J. Linderoth, J. Ostrowski, F. Rossi, and S. Smriglio. Orbital branching. In M. Fischetti and D. Williamson, editors, *Integer Programming and Combinatorial Optimization, Proceedings of the Twelfth International IPCO Conference*, volume 4513 of *LNCS*, pages 106–120. Springer, 2007.

[26] C. Lucet, F. Mendes, and A. Moukrim. Pre-processing and linear-decomposition algorithm to solve the k-colorability problem. In *Proceedings of the Third International Workshop on Experimental and Efficient Algorithms (WEA '04)*, volume 3059 of *LNCS*, pages 315–325. Springer, 2004.

[27] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94(1):71–90, 2002.

[28] F. Margot. Small covering designs by branch-and-cut. *Mathematical Programming*, 94(2–3):207–220, 2003.

[29] F. Margot. Symmetric ILP: Coloring and small integers. *Discrete Optimization*, 4(1):40–62, 2007.

[30] F. Margot. Symmetry in integer linear programming. In M. Jünger, T. Liebling, D. Naddef, G. L. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming 1958–2008*, chapter 17, pages 647–681. Springer, 2010.

[31] B. D. McKay. Practical graph isomorphism. In *Congressus Numerantium*, pages 45–87, 1981.

[32] I. Méndez-Díaz and P. Zabala. A polyhedral approach for graph coloring. *Electronic Notes in Discrete Mathematics*, 7:178–181, 2001.

[33] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.

[34] I. Méndez-Díaz and P. Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2):159–179, 2008.

[35] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[36] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 126(1):147–178, 2011.

[37] J.-F. Puget. On the satisfiability of symmetrical constrained satisfaction problems. In *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems (ISMIS '93)*, pages 350–361. Springer, 1993.

[38] SCIP. Solving Constraint Integer Programs. `http://scip.zib.de`.

[39] E. C. Sewell. An improved algorithm for exact graph coloring. In D. S. Johnson and M. Trick, editors, *Cliques, coloring, and satisfiability. Second DIMACS implementation challenge. Proceedings of a workshop held at DIMACS, 1993*, volume 26 of *Ser. Discrete Math. Theor. Comput. Sci.*, pages 359–373. AMS, DIMACS, 1996.

Cork Constraint Computation Centre and Computer Science Department, University College Cork, Cork, Ireland

E-mail address: `janus@cs.ucc.ie`

Institute for Mathematical Optimization, TU Braunschweig, Pockelsstr. 14, 38106 Braunschweig, Germany

E-mail address: `m.pfetsch@tu-bs.de`