Article

Graph Coloring via Clique Search with Symmetry Breaking

Sándor Szabó [1] and Bogdán Zaválnij [2,*]

[1] Institute of Mathematics and Informatics, University of Pécs, 7622 Pécs, Hungary; sszabo7@hotmail.com
[2] Rényi Institute of Mathematics, 1053 Budapest, Hungary
* Correspondence: bogdan@renyi.hu

Abstract: It is known that the problem of proper coloring of the nodes of a given graph can be reduced to finding cliques in a suitably constructed auxiliary graph. In this work, we explore the possibility of reducing the search space by exploiting the symmetries present in the auxiliary graph. The proposed method can also be used for efficient exact coloring of hyper graphs. We also precondition the auxiliary graph in order to further reduce the search space. We carry out numerical experiments to assess the practicality of these proposals. We solve some hard cases and prove a new lower limit of seven for the mycielski7 graph with the aid of the proposed technique.

Keywords: graph coloring; *k*-clique search; mathematical programming; symmetry breaking

check for updates

Citation: Szabó, S.; Zaválnij, B. Graph Coloring via Clique Search with Symmetry Breaking. *Symmetry* **2022**, *14*, 1574. https://doi.org/10.3390/sym14081574

Academic Editors: Weifan Wang, Min Chen and Yongtang Shi

Received: 9 July 2022
Accepted: 28 July 2022
Published: 30 July 2022

1. Introduction

An implemented algorithm for the maximum clique problem is aptly called a maximum clique solver. Maximum clique solvers have come a long way from the early attempts [1–3] to the fairly capable versions available today. These solvers can be used in industrial settings to solve practical discrete optimization problems.

This work is part of a larger project deploying maximum clique solvers for handling various problems. The authors have already tested this approach in connection with mathematical conjectures [4], hyper graph coloring [5], subgraph isomorphism [6] and scheduling problems [7]. The reader most likely agrees that using a clique solver is similar to using an integer program, a constraint program or a SAT solver. Consequently, it has a space in the toolbox of discrete optimization practitioners.

In this paper, we focus on the optimization problem of proper coloring of the nodes of a given graph with the minimum number of colors by reducing the coloring problem to a clique problem. Naturally, we are not the first to try this approach. There are well-known techniques to convert the coloring problem to a maximum independent set problem. For an example, see [8,9]. We make modifications and improvements to these reductions. For instance, we use the *k*-coloring decision problem instead of the minimization problem. We incorporate symmetry-breaking procedures in the decision version of the problem.

Solving a discrete optimization problem using clique solvers falls into two phases. In the first phase, one sets up an auxiliary graph such that the pertinent features of the problems are coded into this auxiliary graph. In the second phase, one submits the auxiliary graph to the clique solver. The guidelines of constructing an auxiliary graph are as an essential part of this modeling approach as constructing more and more efficient solvers. This phenomenon is well known in the linear programming community.

We see that incorporating one of the symmetry-breaking mechanisms doubles the number of vertices of the auxiliary graph compared to earlier auxiliary graphs without symmetry breaking [9]. Numerical experimentation shows that, in spite of the increase in the nodes of the new auxiliary graph, the search space represented by the search tree associated with the clique search is significantly reduced. The proposed method allowed us to prove a new lower limit 7 for the mycielski7 graph, which, to our knowledge, no other method has been able to do so far. In addition, the proposed method turned out to be the fastest for some cases.

We also introduce kernelization techniques to delete nodes and edges from the auxiliary graph before we submit it to a clique solver. These preconditioning or kernelization methods give a further boost to the clique search.

The structure of the present paper is as follows. First, we quickly describe the basic concepts. Second, we present the auxiliary graph constructions with and without symmetry-breaking methods. Further, we discuss the role of locating and coloring a clique in the originally given graph. Third, we point out that the ideas of this paper can be applied to proper coloring of the vertices of hyper graphs. Fourth, we briefly introduce some kernelization methods. Finally, we present a small toy size example and carry out extended numerical experiments from which we draw conclusions.

2. Definitions

In this paper, we work with graphs with a finite number of nodes and edges that do not have any double edges or loops. In short, we work with finite simple graphs. Let $G = (V, E)$ be a finite simple graph. Here, V is the set of vertices, and E is the set of edges of graph G.

A subset Δ of the nodes of a finite simple graph G is called a k-clique if each two distinct nodes of Δ are adjacent in G. In the special case when Δ has only one element, we still consider it a clique. For each finite simple graph G, there is an integer k such that G has a k-clique but G does not have any $(k+1)$-clique. This well-defined integer k is called the clique number of G and is denoted $\omega(G)$. The optimization problem of computing the clique number of a given graph is a known NP hard problem. The decision problem of deciding if a given graph admits a k-clique for a given integer k, the so-called k-clique problem, is an NP complete problem (see [10,11]).

Subset I of the nodes of finite simple graph G is called an independent set if each two distinct nodes in I are not adjacent in G. If I has only one element, then it is still called an independent set. Note that if I is an independent set in G, then I is a clique in the complement graph of G and vice versa.

We say that coloring of the vertices of G is legal coloring, proper coloring or well coloring if each vertex has exactly one color and adjacent vertices receive distinct colors. The three adjectives for coloring are used freely and interchangeably in the present paper.

For each finite simple graph G, there is an integer k such that the nodes of G can be legally colored using k colors and cannot be legally colored with $(k-1)$ colors. This well-defined integer k is called the chromatic number of G and is denoted $\chi(G)$. It is well-known that the optimization problem of determining the chromatic number of a given graph is NP hard. The k-coloring problem, that is, deciding if the nodes of a given graph admit legal coloring with k colors is a known NP complete problem (see [10,11]). If one uses the numbers $1, \dots, k$ as colors, then coloring of the nodes of G can be described by map $f : V \to \{1, \dots, k\}$. The equation $f(v) = i$ codes the information that node v receives color i. The i-th color class C_i consists of all nodes of G that receive color i. Coloring can also be described by listing the color classes $C_1, \dots, C_k$. Note that color class C_i is an independent set in graph G. Partitioning the set of nodes V of graph G into k independent sets amounts to legally coloring the nodes of G. A nice survey and a taxonomy of the exact node-coloring methods are presented in [12].

3. The Ordered Pairs Auxiliary Graph

The problem of deciding if the n nodes of a given graph G can be legally colored using k colors can be reduced to the problem of deciding if a suitably constructed auxiliary graph Γ with nk nodes contains an n-clique [9]. For the sake of easier reference, we sketch here the construction of the auxiliary graph Γ.

The nodes of Γ are the ordered pairs $[v, c]$, where v is a vertex of G, and c is one of the k colors we use to color the nodes of G. The intended intuitive meaning of pair $[v, c]$ is that node v receives color c. The nodes of the auxiliary graph Γ are also colored. Namely, to node $[v, c]$, we assign v as a color. Two distinct nodes $[v_1, c_1]$, $[v_2, c_2]$ are not adjacent in Γ if they receive the same color, that is, when $v_1 = v_2$. As a consequence, the nodes of Γ are legally colored using n colors. Two distinct nodes $[v_1, c_1]$, $[v_2, c_2]$ are not adjacent in Γ if the unordered pair $\{v_1, v_2\}$ is an edge of G and $c_1 = c_2$.

It was proved in [9] that if the nodes of G can be legally colored using k colors, then the auxiliary graph Γ contains an n-clique. Further, if the auxiliary graph Γ has an n-clique, then the nodes of G can be legally colored using k colors.

We describe a simple observation that can serve as a symmetry breaking tool. We refer to the trick as coloring the nodes of a clique in G. Suppose we are facing the problem of whether the n nodes of a given graph G can be legally colored using the colors $1, \ldots, k$. We try to solve this k-coloring problem by locating an n-clique in the auxiliary graph Γ with nk nodes. Let Δ be an s-clique in the graph G. Note that the nodes $x_1, \ldots, x_s$ of the clique Δ must receive pairwise distinct colors. We may assume that the nodes $x_1, \ldots, x_s$ receive the colors $1, \ldots, s$, respectively, since this is only a matter of rearranging the colors $1, \ldots, k$ among each other in the legal k-coloring of the nodes of G.

The fact that the nodes $x_1, \ldots, x_s$ of the clique Δ receive the colors $1, \ldots, s$, respectively, means that the ordered pairs $[x_1, 1], \ldots, [x_s, s]$, as nodes of the auxiliary graph Γ, must be nodes of any n-clique in Γ that corresponds to legal k-coloring of the nodes of the given graph G. In other words, at this stage of the symmetry-breaking construction, we are fixing the colors of nodes $x_1, \ldots, x_s$ to be the colors $1, \ldots, s$. Naturally, we may restrict the auxiliary graph Γ to the set of common neighbors of the nodes $[x_1, 1], \ldots, [x_s, s]$. After deleting the nodes from the restricted auxiliary graph, we may look for an $(n - s)$-clique in the reduced auxiliary graph in order to locate an n-clique in the original auxiliary graph Γ.

We refer to this symmetry-breaking procedure as coloring the nodes of clique Δ in G. Needless to say, the more nodes clique Δ has, the more efficient our reduction procedure is. Consequently, we try to locate a relatively large clique Δ in G. It is also clear that we may use any greedy procedure to locate clique Δ in G.

4. The Ordered Triplets Auxiliary Graph

In this section, we introduce a new auxiliary graph. The purpose is to augment the previously introduced auxiliary graph with a symmetry-breaking technique, named representative node formulation [13].

Let $G = (V, E)$ be a finite simple graph, and let k be a positive integer. Here, we assume that the nodes of G are labeled with the integers $1, \ldots, n$; that is, we assume that $V = \{1, \ldots, n\}$. Using G and k, we construct a new auxiliary graph $\Gamma = (W, F)$. The nodes of Γ are ordered triples in the form

$$w = [x, y, z], \ \ x \in V, \ \ y \in \{1, \ldots, k\}, \ \ z \in \{1, 2\}.$$

The number z is called the type of the triple. We talk about Type-1 and Type-2 nodes depending on whether $z = 1$ or $z = 2$. The intended intuitive meaning of a Type-1 node $w = [x, y, z]$ of Γ is that node x of G receives color y. The intended intuitive meaning of a Type-2 node $w = [x, y, z]$ of Γ is that node x of G receives color y, and, in addition, x is the first element of the y-th color class. In other words, if node x' of G receives color y, then $x \leq x'$ must hold. We point out that it makes sense to talk about first, second and last elements of a color class as the nodes of graph G admit the natural ordering of the numbers $1, \ldots, n$.

We color the nodes of Γ in the following manner. If $w = [x, y, z]$ is a Type-1 node of Γ, then we assign x as a color to w. If $w = [x, y, z]$ is a Type-2 node of Γ, then we assign $(n + y)$ as a color to w. The auxiliary graph Γ has $(2nk)$ nodes, and the nodes are colored with $(n + k)$ colors.

We describe the adjacency in Γ. Let

$$w_1 = [x_1, y_1, z_1], \quad w_2 = [x_2, y_2, z_2]$$

be distinct nodes of Γ. As a starting point, we connect each two distinct nodes of Γ with an edge; that is, at the beginning of the construction, Γ is a complete graph.

If nodes w_1, w_2 receive the same color in Γ, then we make them nonadjacent in Γ by deleting the edge connecting them. For the remaining part of the construction, we assume that w_1, w_2 do not receive the same color. We distinguish cases depending on the types of w_1, w_2.

Assume first that w_1, w_2 are both Type-1 nodes of Γ; that is, $z_1 = z_2 = 1$. If x_1, x_2 are adjacent vertices in G, and $y_1 = y_2$, then w_1, w_2 are not adjacent in Γ, and we delete the corresponding edge from Γ.

Assume next that w_1, w_2 are both Type-2 nodes of Γ; that is, $z_1 = z_2 = 2$. If $x_1 \geq x_2$ and $y_1 \leq y_2$, then w_1, w_2 are not adjacent in Γ, and we delete the corresponding edge from Γ. If $x_1 \leq x_2$ and $y_1 \geq y_2$, then w_1, w_2 are not adjacent in Γ, and we delete the corresponding edge from Γ.

Finally, assume that w_1, w_2 are Type-1, Type-2 nodes of Γ, respectively; that is, $z_1 = 1$, $z_2 = 2$. If $x_1 < x_2$ and $y_1 \geq y_2$, then w_1, w_2 are not adjacent in Γ, and we delete the corresponding edge from Γ. Further, if $x_1 = x_2$ and $y_1 \neq y_2$, then w_1, w_2 are not adjacent in Γ, and we delete the corresponding edge from Γ.

Lemma 1. *If the nodes of G can be legally colored using k colors, then the auxiliary graph Γ contains an $(n+k)$-clique.*

Proof. Suppose that the nodes of G are legally colored using the colors $1, \ldots, k$, and $C_1, \ldots, C_k$ are the colors classes of the nodes. We set $\mu(i) = \min(C_i)$ for each i, $1 \leq i \leq k$. We may assume that $\mu(1) < \cdots < \mu(k)$ holds since this is only a matter of exchanging the colors $1, \ldots, k$ among each other. In other words, we may assume that the function $\mu : \{1, \ldots, k\} \to \{1, \ldots, n\}$ is a strictly increasing function.

Let $f : V \to \{1, \ldots, k\}$ be the map that describe the coloring of the nodes of G. (Here, $V = \{1, \ldots, n\}$ is the set of nodes of G). Note that $f(\mu(i)) = i$ holds for each i, $1 \leq i \leq k$. We claim that the triples

$$[r, f(r), 1], \ 1 \leq r \leq n, \qquad [\mu(s), s, 2], \ 1 \leq s \leq k$$

are the nodes of an $(n+k)$-clique in Γ. In plain English, we claim any two of these $(n+k)$ nodes are pairwise adjacent.

In order to prove the claim, we verify the following. The distinct nodes

$$w_1 = [i, f(i), 1], \quad w_2 = [j, f(j), 1] \tag{1}$$

of the auxiliary graph Γ are adjacent for each i, j, $1 \leq i < j \leq n$. In plain English, the first (n) nodes are pairwise adjacent. The distinct nodes

$$w_1 = [\mu(i), i, 2], \quad w_2 = [\mu(j), j, 2] \tag{2}$$

of the auxiliary graph Γ are adjacent for each i, j, $1 \leq i < j \leq k$. In plain English, the last (k) nodes are pairwise adjacent. The distinct nodes

$$w_1 = [i, f(i), 1], \quad w_2 = [\mu(j), j, 2] \tag{3}$$

of the auxiliary graph Γ are adjacent for each i, j, $1 \leq i \leq n$, $1 \leq j \leq k$. In plain English, any of the first n nodes are adjacent to any of the last k nodes.

We first consider (1). When the unordered pair $\{i, j\}$ is not an edge of G, then by the definition of Γ, the nodes w_1, w_2 are adjacent in Γ. When the unordered pair $\{i, j\}$ is an edge of G, then the inequality $i < j$ implies $f(i) \neq f(j)$ as f defines a legal coloring of the nodes of G. The definition of Γ gives that nodes w_1, w_2 are adjacent in Γ.

Next, we consider (2). In this situation, $i < j$ implies $\mu(i) < \mu(j)$. By the definition of Γ, it follows that nodes w_1, w_2 are adjacent in Γ.

Finally, we consider (3). If $i < \mu(j)$, then node i of G cannot be an element of the color class C_j, and so it follows that $f(j) > f(i)$. Using $f(j) = j$, we get that $i < \mu(j)$ and $f(i) < j$ hold. The definition of Γ gives that nodes w_1, w_2 are adjacent in Γ.

If $i = \mu(j)$, then $f(i) = f(\mu(j)) = j$. Therefore $i = \mu(j)$ and $f(i) = j$ hold. The definition of Γ gives that nodes w_1, w_2 are adjacent in Γ. □

Lemma 2. *If the auxiliary graph Γ contains an $(n+k)$-clique, then the nodes of G can be legally colored using k colors.*

Proof. Let Δ be an $(n+k)$-clique in Γ. The nodes of Γ are legally colored with $(n+k)$ colors. It follows that the $(n+k)$ nodes of Δ receive all the possible $(n+k)$ colors. Therefore, the nodes of Δ are in the following forms.

$$[x_r, y_r, 1],\ \ 1 \le r \le n, \quad [x_s, y_s, 2],\ \ 1 \le s \le k$$

The first n nodes of Δ receive the colors $1, \ldots, n$, and the last k nodes of Δ receive the colors $n+1, \ldots, n+k$, respectively. The color of node $[x_r, y_r, 1]$ in Γ is x_r. Thus elements $x_1, \ldots, x_n$ form a rearrangement of elements $1, \ldots, n$. In other words, $V = \{1, \ldots, n\} = \{x_1, \ldots, x_n\}$. Let $f : V \to \{1, \ldots, k\}$ be the map defined by $f(x_r) = y_r$. We claim that the coloring of the nodes of G described by the map f is a legal coloring of the nodes of G. In order to verify the claim, assume on the contrary that the unordered pair $\{x_i, x_j\}$ is an edge of G, and $y_i = f(x_i) = f(x_j) = y_j$. By the definition of Γ, the nodes $w_1 = [x_i, y_i, 1]$, $w_2 = [x_j, y_j, 1]$ are not adjacent in Γ. On the other hand, the nodes w_1, w_2 are distinct nodes of the clique Δ, and they must be adjacent in Γ. □

The symmetry-breaking trick of coloring the nodes of a clique Δ in the given graph G can be applied in connection with triplet auxiliary graphs as well.

Find an s-clique Δ and color the nodes $x_1, \ldots, x_s$ with the colors $1, \ldots, s$, respectively. Note that we may rename nodes $1, \ldots, n$ of graph G such that nodes $x_1, \ldots, x_s$ of Δ are identical to nodes $1, \ldots, s$ of G. As a next step, we may reduce the auxiliary graph. First, we restrict the auxiliary graph to the common neighbors of nodes $1, \ldots, s$. Next, we delete nodes $1, \ldots, s$ from the reduced auxiliary graph.

5. Chromatic Number via Maximum Clique

In connection with a given graph G, we use four auxiliary graphs Γ^1, Γ^2, Γ^3, Γ^4 systematically. Here Γ^1, Γ^3 are the pair and triplet auxiliary graphs, respectively, while Γ^2, Γ^4 are reduced graphs we get after coloring the nodes of a clique in the given graph G. For each of the above auxiliary graphs, add some extra nodes and construct a new auxiliary graph $(\Gamma^i)'$. Feeding graph $(\Gamma^i)'$ into a maximum clique problem solver, we determine the size of a maximum clique. Using the clique number of $(\Gamma^i)'$, we determine the chromatic number $\chi(G)$ of the given graph G.

As a first step, we establish a lower bound B_l and an upper bound B_u for the chromatic number $\chi(G)$ of G. For the sake of definiteness and for the sake of simplicity using a greedy procedure, we locate a clique in graph G. The size of this clique provides the lower bound B_l for $\chi(G)$. A greedy-coloring procedure applied to the nodes of G gives the upper bound B_u for $\chi(G)$. Of course, the reader is free to use more-sophisticated methods for finding the bounds B_l, B_u.

As a second step, we set $k = B_u - 1$ and construct the $(\Gamma^1)'$, $(\Gamma^2)'$, $(\Gamma^3)'$, $(\Gamma^4)'$ graphs. Here is the construction. We add $t = k - B_l$ extra nodes $a_1, a_2, \ldots, a_t$ to the graph Γ^i. The intended intuitive meaning of node a_j is that color $B_l + j$ is not used in the coloring of the nodes of graph G.

As a third step, we add new edges to the graph $(\Gamma^i)'$:

- Connect a_p and a_q with an edge for each p, q;
- Connect a_p to the ordered pair $[x, y]$ with an edge for each x and $y < B_l + p$ (for Γ^1, Γ^3);

- Connect a_p to the ordered triplet $[x, y, z]$ with an edge for each x and $y < B_l + p$, $z \in \{1, 2\}$ (for Γ^2, Γ^4).

After computing the clique number ω_a of the newly constructed auxiliary graph, we can calculate the chromatic number of the given graph G.

For a Type-1 auxiliary graph, if $\omega_a < n$, then $\chi(G) = B_u$. Else, $\chi(G) = B_u - 1 - (\omega_a - n)$.

For a Type-2 auxiliary graph, if $\omega_a < n - s$, then $\chi(G) = B_u$. Else $\chi(G) = B_u - 1 - (\omega_a - n + s)$ (Remember, s is the size of the clique Δ we located and colored in G).

For a Type-3 auxiliary graph, if $\omega_a < n + k$ then $\chi(G) = B_u$. Else $\chi(G) = B_u - 1 - (\omega_a - n - k)$.

For a Type-4 auxiliary graph, if $\omega_a < n + k - 2s$, then $\chi(G) = B_u$. Else $\chi(G) = B_u - 1 - (\omega_a - n - k + 2s)$.

6. Coloring Hyper Graphs

The purpose of this section is to extend the symmetry-breaking methods from ordinary graphs to hyper graphs. In our setting, a hyper graph H is an ordered pair (V, E), where V is the set of vertices of H, and E is a family of subsets of V. We refer to the elements of E as hyper edges of H. We do not assume that the members of E all have the same number of elements. Coloring the vertices of H is called legal, proper or well coloring if each vertex receives exactly one color and vertices belonging to a hyper edge do not all receive the same color. In other words, monochromatic edges are not allowed.

In [5], the authors described how well coloring of the vertices of a given hyper graph can be reduced to a clique search in a suitably constructed ordinary graph. The reader can notice that if the vertices of a hyper graph are well-colored using k colors, then the colors can be permuted among each other and the resulting vertex coloring of the hyper graph remains a well coloring. In this way, one particular well coloring of the vertices leads to a large number of new well colorings of the hyper graph. Namely, we end up with $(k!)$ new colorings. Here, k is the number of colors used in the coloring.

In an earlier part of this paper, we saw that in the special case of coloring the vertices of an ordinary graph, we may reduce the size of the search space of the associated clique search by introducing some symmetry-breaking tools. In what follows, we point out that ideas similar to those used for ordinary graphs can be used for symmetry-breaking in the case of hyper graphs.

In [5], an ordinary auxiliary graph Γ was assigned to a given hypergraph $H = (V, E)$. In the course of the construction of Γ, the hyper edges of H are divided into pairwise disjoint subsets, so-called tiles. Depending on the number of tiles, one computes a number r. The essential property of this assignment is that a well coloring of the vertices of H using k colors corresponds to a clique of size r in the auxiliary graph Γ. Conversely, a clique of size r in Γ corresponds to a well coloring of the vertices of H using k colors.

Using graph Γ, we construct a new auxiliary graph Γ'. Here again, we use the representative node technique by adding new nodes to the auxiliary graph Γ to get Γ' Suppose U is the set of vertices of Γ, and the elements of V are listed in a fixed order during construction. We consider the ordered pairs (v, c), where $v \in V$, and $1 \leq c \leq k$. The intended intuitive meaning of pair (v, c) is that the vertex v of the hyper graph H receives color c, and, in addition, node v is the first element in its color class. Because of the ordering of the vertices of H, it makes sense to talk about the first element of a color class.

Let $X = \{(v, c) : \ v \in V,\ 1 \leq c \leq k\}$. The set $W = X \cup U$ is the set of nodes of the new auxiliary graph Γ'. Note that sets X and U are disjointed.

Two distinct elements w_1 and w_2 of W are adjacent in Γ' whenever they are adjacent in Γ.

Two distinct elements (v_1, c_1) and (v_2, c_2) of X cannot be adjacent in Γ' if $v_1 \leq v_2$ and $c_1 \geq c_2$.

Finally, let us turn to the case of when vertex $u \in U$ and vertex $(v, c) \in X$ are adjacent in Γ'. In the definition of the auxiliary graph Γ, vertex u carries an intuitive meaning regarding colors that certain well-defined nodes of hyper graph H receive. We connect the nodes u and (v, c) with an edge in Γ' if the color assignments specified in the definition of Γ do not violate the fact that node v receives color c.

The essential property of the new auxiliary graph Γ' is the following: A well coloring of the nodes of hyper graph H using k colors corresponds to a clique of size $(r + k)$ in Γ'. Conversely, a clique of size $(r + k)$ in Γ' corresponds to a well coloring of the nodes of H using k colors.

With the same technique, we can add symmetry-breaking to other types of hyper graph colorings, such as rainbow coloring or C–D coloring.

7. Kernelization

Let us denote the size of the clique we are looking for in the Γ auxiliary graph by z for the rest of this paper.

In the ordered-pairs auxiliary graph Γ^1 associated with G, we are looking for an n-clique, that is, $z = n$. For Γ^2, where we color the nodes of clique Δ, $z = n - |\Delta|$. For Γ^3, the triplets auxiliary graph, $z = n + k$, and for Γ^4, $z = n + k - 2|\Delta|$.

It is reasonable to try to kernelize the auxiliary graph Γ.

Let v be a vertex of Γ, and let H be the subgraph of Γ induced by the set $N(v)$. We showed that the nodes of Γ are legally colored using z colors (Though, this is not a perfect graph). As a consequence, the nodes of H are colored. We call the number of colors of the nodes of H the color-degree of node v. Let $e = \{u, v\}$ be an edge of Γ, and let H be the subgraph of Γ induced by the set $[N(u) \cap N(v)]$. The number of colors of the nodes of H is called the color-degree of edge $e = \{u, v\}$.

Lemma 3. *A node whose color degree is less than $(z - 1)$ can be deleted from Γ when we are looking for a z-clique in Γ. An edge whose color degree is less than $(z - 2)$ can be deleted from Γ when we are looking for a z-clique in Γ.*

Let u and v be nonadjacent nodes in Γ, and $N(u) \subseteq N(v)$; then, we say node v dominates node u. Let $e = \{u, v\}$, $f = \{v, w\}$ be edges of Γ. If u, w are not adjacent in Γ, and $[N(u) \cap N(v)] \subseteq [N(v) \cap N(w)]$, then we say that edge f dominates edge e. The following result has been proved in [14].

Lemma 4. *If node v dominates node u, then u maybe be deleted from Γ when we are looking for a z-clique in Γ. If edge f dominates edge e, then e maybe deleted from Γ when we are looking for a z-clique in Γ.*

Obviously a full-degree node can be deleted from the graph when we are looking for the clique number of the graph. We should keep in mind this deletion reduces the clique number by one. For the sake of easier reference, we state this result as a lemma.

Lemma 5. *A full-degree node can be deleted from Γ when we are looking for a z-clique in Γ.*

We listed here only the simplest kernelization methods that can be applied for kernelization of a k-partite graph. For more extended methods, including β-transformation, specialized struction and the concept of black–red edges, see [7].

A possible kernel of the graph Γ is constructed from Γ by deleting nodes and edges by repeatedly applying Lemmas 3–5 and by other methods from [7]. Our experiments show that these reductions reduce the graphs quite well, and repeatedly applying them helps a lot.

As for the maximum clique reformulation in Section 5, the situation is a bit more complicated. One would like to first reduce the auxiliary graph by the previous rules and to add the extra nodes. However, there are important differences. Lemma 4—in contrast to the other two lemmas—can delete z-cliques from Γ, ensuring that at least one z-clique remains. However, for the maximum clique reformulation, this behavior breaks the mathematical result. Thus, we can first reduce the graph by Lemmas 3 and 5, and *afterward* add the extra nodes. However, we can apply Lemma 4 only after adding these nodes, and we cannot use the other two lemmas later, making this approach less suitable for kernelization.

Overall, our experiments proved that, although this reformulation is interesting, solving decision problems is more efficient after all.

8. A Small-Size Toy Example

In this section, we work out a small-size toy example in detail to illustrate our definitions and constructions.

Example 1. *Let us consider the graph $G = (V, E)$ given by its adjacency matrix in Table 1. The graph has four vertices* 1, 2, 3, 4 *and four edges* $\{1,2\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$*; that is,*

$$V = \{1,2,3,4\}, \quad E = \{\{1,2\},\{1,4\},\{2,3\},\{2,4\}\}.$$

Figure 1 depicts a possible geometric representation of the graph. We are asking if the nodes of G can be colored legally using three colors.

Table 1. The adjacency matrix of the graph G in Example 1.

	1	2	3	4
1	×	●		●
2	●	×	●	●
3		●	×	
4	●	●		×

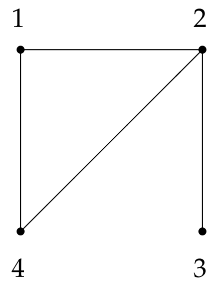

Figure 1. A possible geometric representation of graph G in Example 1.

In this particular case, $n = 4$, $k = 3$, and the triplet auxiliary graph Γ has $\binom{2}{nk} = 24$ vertices. The vertices of Γ are listed in Table 2, including the types and colors assigned to them in Γ. The adjacency matrix of Γ is in Table 3. The auxiliary graph has 144 edges. The nodes of graph Γ are well-colored using 7 colors, and we are looking for a 7-clique in Γ.

By inspecting the adjacency matrix, the reader may notice that the color-degree of node $[4,1,2]$ is equal to 1, so Lemma 3 gives that this node can be deleted from Γ. Next, one may notice that the color degrees of nodes $[1,2,1]$, $[1,3,1]$ are small, and so these nodes can also be deleted. Continuing in this way, we end up with a reduced version of Γ, the adjacency matrix of which is enclosed in Table 4.

Table 2. The nodes of the ordered triplet auxiliary graph Γ in Example 1.

Name	Triple	Type	Color	Name	Triple	Type	Color
1	[1, 1, 1]	1	1	13	[1,1,2]	2	5
2	[1, 2, 1]	1	1	14	[2, 1, 2]	2	5
3	[1, 3, 1]	1	1	15	[3, 1, 2]	2	5
4	[2, 1, 1]	1	2	16	[4, 1, 2]	2	5
5	[2, 2, 1]	1	2	17	[1, 2, 2]	2	6
6	[2, 3, 1]	1	2	18	[2, 2, 2]	2	6
7	[3, 1, 1]	1	3	19	[3, 2, 2]	2	6
8	[3, 2, 1]	1	3	20	[4, 2, 2]	2	6
9	[3, 3, 1]	1	3	21	[1, 3, 2]	2	7
10	[4, 1, 1]	1	4	22	[2, 3, 2]	2	7
11	[4, 2, 1]	1	4	23	[3, 3, 2]	2	7
12	[4, 3, 1]	1	4	24	[4, 3, 2]	2	7

Table 3. The adjacency matrix of the triplet auxiliary graph in Example 1.

	111	121	131	211	221	231	311	321	331	411	421	431	112	212	312	412	122	222	322	422	132	232	332	432
111	×				●	●	●	●	●		●	●	●					●	●	●		●	●	●
121		×		●		●	●	●	●	●		●					●					●	●	●
131			×	●	●		●	●	●	●	●										●			
211		●	●	×				●	●		●	●	●	●			●		●	●	●		●	●
221	●		●		×		●		●	●		●	●				●	●			●		●	●
231	●	●				×	●	●		●	●		●				●				●	●		
311	●	●	●		●	●	×			●	●	●	●	●	●		●	●		●	●	●		●
321	●	●	●	●		●		×		●	●	●	●	●			●	●	●		●	●		●
331	●	●	●	●	●				×	●	●	●	●	●			●	●			●	●	●	
411		●	●		●	●	●	●	●	×			●	●	●	●	●	●	●		●	●	●	
421	●		●	●		●	●	●	●		×		●	●	●		●	●	●	●	●	●	●	
431	●	●		●	●		●	●	●			×	●	●	●		●	●	●		●	●	●	●
112	●			●	●	●	●	●	●	●	●	●	×					●	●	●		●	●	●
212				●			●	●	●	●	●	●		×					●	●			●	●
312							●			●	●	●			×					●				●
412										●						×								
122		●		●	●	●	●	●	●	●	●	●					×					●	●	●
222	●				●		●	●	●	●	●	●	●					×					●	●
322	●			●				●		●	●	●	●	●					×					●
422	●			●			●				●		●	●	●					×				
132			●	●	●	●	●	●	●	●	●	●									×			
232	●	●				●	●	●	●	●	●	●	●				●					×		
332	●	●		●	●				●	●	●	●	●	●			●	●					×	
432	●	●		●	●		●	●				●	●	●	●		●	●	●					×

Table 4. The adjacency matrix of the kernelized triplet auxiliary graph in Example 1.

	1	2	3	3	4	1	2	3	4
	1	2	1	3	3	1	2	3	3
	1	1	1	1	1	2	2	2	2
111	×	●	●	●	●	●	●	●	●
221	●	×	●	●	●	●	●	●	●
311	●	●	×		●	●	●		●
331	●	●		×	●	●	●	●	
431	●	●	●	●	×	●	●	●	●
112	●	●	●	●	●	×	●	●	●
222	●	●	●	●	●	●	×	●	●
332	●	●		●	●	●	●	×	
432	●	●	●		●	●	●		×

The reduced graph has five full-degree nodes; thus, one can draw a geometric representation of the graph easily. A possible geometric representation of the reduced auxiliary graph is depicted in Figure 2. We can locate two 7-cliques in the reduced graph. The nodes of these cliques are the following:

$$[1,1,1],[1,1,2],[2,2,1],[2,2,2],[4,3,1],[3,3,1],[3,3,2],$$

$$[1,1,1],[1,1,2],[2,2,1],[2,2,2],[4,3,1],[3,1,1],[4,3,2].$$

Using the definition of the nodes, we can read two distinct well colorings of the nodes of the originally given graph G. These colorings of the nodes of G are listed in Table 5.

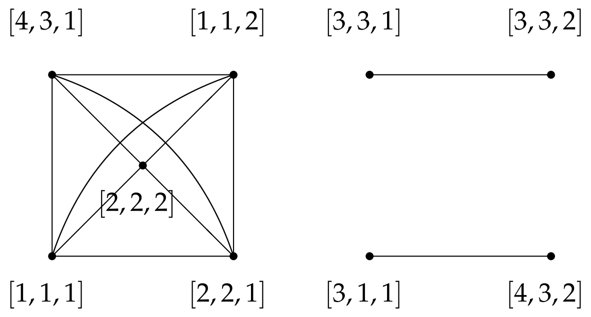

Figure 2. A graphical representation of the kernelized auxiliary graph in Example 1. Each of the five vertices on the right is adjacent to each of the four vertices on the left. To avoid an overly cluttered picture, these twenty edges are not included.

Table 5. The colors of graph G in Example 1.

node	1	2	3	4	node	1	2	3	4
color	1	2	1	3	color	1	2	3	3

9. Numerical Experiments

The present paper describes some new ideas for exact graph coloring. Although the results are not mature, and the present work is rather preliminary, we would like to demonstrate the effectiveness of this approach by comparing our results to results of other well-known graph coloring applications. We restrict ourselves to graph coloring, as the literature on exact hyper graph coloring is scarce. We also decided to exclude reformulation when maximum clique of the auxiliary graph gives us the minimum coloring of the graph as detailed in Section 5. The reason behind this is that we have less-effective preconditioning tools for maximum clique search compared to preconditioning on the k-clique search in k-partite graphs.

For the calculations, we are using 58 graphs from the standard clique-coloringbenchmark instances http://mat.gsia.cmu.edu/COLOR04 (accessed on 15 June 2022). All calculations were done single-threaded on a Linux OS computer with two AMD EPYC 7643 processors and 1 TB of RAM; the boost was switched off, and thus it ran exactly at 2.2 GHz. We used gcc v12.1 with the switch settings `-O3 -arch=znver3`. The dfmax times on this computer were 0.00, 0.02, 0,12, 0.70 and 2.59 s for r100, r200.5, r300.5, r400.5 and r500.5, respectively.

9.1. Setting Up the Testbed

Before making comparative measurements, one needs to make a decision about the different approaches listed above. Apart from setting aside maximum-clique reformulation, as preliminary tests showed its inferiority due to less-efficient kernelization, there are still four different reformulations. Further, preconditioning takes its toll, while it reduces the graph, the reduction may help too little, and the time for preconditioning may be too long. First, then, we need to compare the four reformulations each with or without preconditioning. For this purpose, we choose three moderately hard example cases. Note that because we tune to hard cases, we may be a bit less efficient for other, easy cases. The results are summarized in Table 6. In the table, we first list the name of the graph for the test, the size (number of nodes) of the graph and the chromatic number of the graph. For testing, we set $k = \chi(G) - 1$ to prove that the graph cannot be colored with one fewer colors than the chromatic number. We find a greedy maximal clique by using maximum-neighbors heuristics, and we list the size of the clique. We construct Γ^1, Γ^2, Γ^3, Γ^4. We list the size of Γ and the z number, that is, the size of the clique should one prove not to be present in the graph according to Section 7. Then, we solve the problem by using our clique search program for k-clique from [15], and we list the running time in seconds.

Table 6. Preliminary tests for three graphs: tl, time limit of 24 h exceeded. We denote with boldface the best time.

	$\mathbf{\Gamma^1}$	$\mathbf{\Gamma^2}$	$\mathbf{\Gamma^3}$	$\mathbf{\Gamma^4}$
myciel5, size: 47, $\chi = 6$				
greedy clique size: 2				
size of Γ	235	192	470	327
z number	47	45	52	48
time of exact z-clique, no precond (s)	171.13	13.12	3.14	12.97
time of preconditioning (s)	0.25	0.21	0.45	0.58
Γ size after preconditioning	235	211	258	219
z number after preconditioning	47	41	39	22
time of exact z-clique with precond (s)	167.93	1.18	3.23	0.11
full time to solve the problem (s)	168.18	1.39	3.68	**0.69**
2-FullIns_4, size: 212, $\chi = 6$				
greedy clique size: 4				
size of Γ	1060	832	2120	1040
z number	212	208	217	209
exact z-clique, no precond (s)	tl	tl	tl	tl
time of preconditioning (s)	86.37	5.98	203.07	3.66
Γ size after preconditioning	1060	0	1187	0
z number aft precond.	211		167	
time of exact z-clique with precond (s)	tl		tl	
full time to solve the problem (s)	tl	5.98	tl	**3.66**

Table 6. *Cont.*

	Γ^1	Γ^2	Γ^3	Γ^4
1-Insertions_4, size: 67, $\chi = 5$				
greedy clique size: 2				
size of Γ	268	232	536	362
z number	67	65	71	67
exact z-clique, no precond (s)	tl	8338.49	1683.63	8638.34
time of preconditioning (s)	0.71	0.47	1.58	1.32
Γ size after preconditioning	296	257	314	337
z number aft precondn	60	48	50	35
time of exact z-clique with precond (s)	tl	487.66	3416.41	303.82
full time to solve the problem (s)	tl	488.13	3417.99	**305.14**

Next, we perform extended preconditioning according to [7], and we list the running time. The size of the reduced graph and the new z-number is listed; because of preconditioning, the number of color classes and the clique size one must find in the preconditioned graph changes. Note that preconditioning allows slight increases to the graph, so in some cases, the reduced graph may be bigger. In some cases, preconditioning reduces the graph to size 0 and thus solves the problem by itself. Next, we run the k-clique search on the reduced graph, listing the running times. Finally, we list the overall running time to solve the problem. The term "tl" denotes running time over the time limit, which was 24 h in this case.

Although intermediate results from these examples are quite mixed, the final results are clear. Reformulation to Γ^4 with preconditioning is the right method, which obviously was the expected outcome. So for the experiments, we use this method.

9.2. Extended Tests

We compare our approach to four different state-of-the-art programs. The first, by Zhou et al. [16], is a backtracking algorithm aided by a SAT solver. The second and third are by Cornaz et al. [17]. The second uses a smart reformulation to LP and solves the problem with column generation. The third uses representative form reformulation—the same symmetry-breaking technique we use in the present paper. The fourth, by Malaguti et al. [18], uses a pure LP approach and solves the problem using a Branch-and-Price algorithm and column generation. Note that the first method is not fully comparable to the other three in terms of speed, as the first can solve instances that the others cannot and vice versa. Being fully combinatorial, it is not surprising that our method is closer to the first program by Zhou et al. The program from Zhou et al. can be downloaded, so we reproduced the calculations on our computer with a recompiled program. The programs from Cornaz et al. and from Malaguti et al. are not available, so we ended up using the results from the original papers. Because of the differences amongst the computers, in order to compare the results, we set the time limit of our runs to 3 h 45 m—that is, 13,500 s, which roughly equals the scaled time limit from [17]. Further, while using data from [17], we scaled those results by dividing the running times by 1.74. The data from [18] was scaled by dividing by 2.7.

The steps of the calculations follow:

1. We located a possibly big clique Δ by a simple algorithm (choosing the biggest degree nodes iteratively)—this is our lower bound (lb);
2. We colored the graph by a heuristic algorithm (DSatur by Brelaz [19] and iterated recoloring by Culberson [20]) —this gives us an upper bound (ub);
3. We constructed the auxiliary graph Γ^4 using $k \in \{ub - 1, \dots, lb\}$; that is, for all possible k between the bounds in a top-down approach. However, we stopped if we reached a conclusive result;
4. We applied the above-described kernelization steps on the resulting graphs;

5. We performed a z-clique search on the kernelized auxiliary graphs using our k-clique program [15].

Locating a clique, coloring the graph and constructing the auxiliary graph for most benchmark problems was fast—under 0.1 seconds in the presented cases. The full running time was dominated by the time for preconditioning and the time for the z-clique search in the auxiliary graph. Kernelization of some graphs could delete every edge and node from that graph eventually, which resulted in an empty graph. In this case, we did not need to run the clique search, as we already knew that there was no z-clique in the graph. The results are summarized in Table 7. The columns correspond to the above-listed state-of-the-art programs, while the last column is the program of the present paper.

Table 7. Running times in seconds for benchmark graphs—finding χ and proving its optimality: tl, time limit of 3 h 45 m reached; ND, no data. We denote with boldface if a solution is the best or if there is a draw between the first two places.

	$\|V\|$	χ	**Zhou cdclGCP**	**Cornaz MWSS**	**Cornaz REPf**	**Malaguti MMT-BP**	**Szabo, Zavalnij**
1-Insertions_4	67	5	**28**	tl	8650	tl	305
1-Insertions_5	202	?	tl	tl	tl	tl	tl
2-Insertion_4	149	4	tl	tl	tl	tl	tl
2-Insertion_5	597	?	tl	ND	ND	tl	tl
3-Insertion_3	56	4	**0**	ND	ND	tl	**0**
3-Insertion_4	281	?	tl	ND	ND	tl	tl
3-Insertion_5	1406	?	tl	ND	ND	tl	tl
4-Insertions_3	79	4	2	tl	7795	tl	**1**
4-Insertion_4	475	?	tl	ND	ND	tl	tl
1-FullIns_4	93	5	**0**	10	3	tl	**0**
1-FullIns_5	282	6	tl	tl	tl	tl	**12,399**
2-FullIns_3	52	5	0	0	0	1	0
2-FullIns_4	212	6	tl	9208	67	tl	**4**
2-FullIns_5	852	7	tl	ND	ND	tl	tl
3-FullIns_3	80	6	tl	1	**0**	1	**0**
3-FullIns_4	405	7	tl	ND	ND	tl	**17**
4-FullIns_3	114	7	tl	ND	ND	1	**0**
4-Fullins_4	690	8	tl	ND	ND	tl	tl
4-Fullins_5	4146	?	tl	ND	ND	tl	tl
5-FullIns_3	154	8	tl	9	**1**	2	**1**
5-FullIns_4	1085	?	tl	ND	ND	tl	tl
anna	138	11	0	2	0	1	0
david	87	11	0	0	0	0	0
huck	74	11	tl	0	0	0	0
jean	80	10	0	0	0	0	0
myciel3	11	4	0	0	0	3	0
myciel4	23	5	0	0	0	71	0
myciel5	47	6	**0**	87	35	tl	1
myciel6	95	7	**1188**	tl	tl	tl	1367
myciel7	191	8	tl	tl	tl	tl	tl
DSJC125.1	125	5	**0**	9951	38	53	40
DSJC125.5	125	17	tl	tl	tl	**6685**	tl
DSJC125.9	125	44	tl	**0**	**0**	1443	tl
DSJC250.1	250	?	tl	tl	tl	tl	tl
DSJC250.5	250	?	tl	tl	tl	tl	tl
DSJC250.9	250	72	tl	**1118**	1388	tl	tl
DSJR500.1c	500	85	tl	**1**	**1**	107	tl
games120	120	9	tl	3	**0**	**0**	23

Table 7. *Cont.*

	$\lvert V \rvert$	χ	**Zhou cdclGCP**	**Cornaz MWSS**	**Cornaz REPf**	**Malaguti MMT-BP**	**Szabo, Zavalnij**
queen8_8	64	9	2	87	20	**1**	14
queen9_9	81	9	200	tl	tl	**14**	12,783
queen10_10	100	11	tl	tl	tl	**254**	tl
queen11_11	121	12	tl	tl	tl	**691**	tl
queen12_12	144		tl	tl	tl	tl	tl
miles250	250	8	0	1	0	2	0
miles500	500	20	0	1	0	1	0
miles750	750	31	0	0	1	0	0
miles1000	1000	42	tl	**0**	2	**0**	1
miles 1500	1500	73	3	0	1	0	0
school1	385	14	0	ND	ND	0	0
school1_nsh	352	14	**0**	ND	ND	6	**0**
le450_5a	450	5	0	ND	ND	0	0
le450_5b	450	5	0	ND	ND	0	0
le450_5c	450	5	0	ND	ND	0	0
le450_5d	450	5	0	ND	ND	0	0
le450_15a	450	15	tl	ND	ND	**0**	tl
le450_15b	450	15	tl	ND	ND	**0**	tl
le450_15c	450	15	tl	ND	ND	**1**	tl
le450_15d	450	15	tl	ND	ND	**1**	tl

The results are quite interesting. The miles graphs, with the exception of miles1000, could be solved by our method simply because we could locate a clique in them of the same size as the numbers of colors we could color the graph with. In these cases, we need not run our clique search at all. Some other graphs, for example miles1000, could be solved by only preconditioning. Some time results of the Zhou et al. program were not consistent with their paper (the queen9_9 is much better, while 1-FullIns_5, 2-FullIns_4, 3-FullIns_3, 5-FullIns_3, huck and games120 are much worse), so this problem needs more investigation. For other instances, our approach is quite close to the Zhou et al. program.

If we calculate the first places from Table 7, that is, the data denoted with boldface, including draws between the first two application, then the results are as follows. First place is Malaguti MMT-BP, with 11 first places. Second is our approach, with 10 first places. Third is Zhou cdclGCP, with 7 first places. Programs from Cornaz gained 4 and 3 first places. If we calculate the number of instances solved within the time limit, then the program Zhou cdclGCP and our approach both solved 34 out of 58 instances, and Zhou cdclGCP solved 25 out of 58 instances. Programs from Cornaz et al. solved 23 and 25 out of 37 instances.

Not listed in the table, we also solved the mycielski7 graph for $k = 6$ in 19,161 s; which proves that there is no proper 6 coloring of the graph mycielski7. This result improves the previously calculated lower limit for this instance to 7, while the usual good lower bound is 5, and the previous best is 6. The program from Zhou et al. could not solve the same question in 48 h.

Note though, the downside of our approach. There are some instances, not listed in the table, that our approach simply could not handle. These are the more dense instances with a higher chromatic number. In these cases, our reformulation constructs a huge graph, as the basic size of our graph is (nk). These auxiliary graphs can be of size ten thousand or more, and thus are beyond our approach. Basically, if the Γ graph is over 3000 nodes, and after preconditioning, the resulting graph is over 1000 nodes, then our approach usually cannot solve the instance. One may conclude that in those cases the auxiliary graphs from Cornaz et al., whose sizes are small in these cases and big in cases favoring our approach, are better for solving such problems. This orthogonal property may lead to a portfolio approach, in which, after inspecting the graph, one can choose between the two methods.

10. Conclusions

Our main goal in the present paper was to elaborate on the question if clique search can be used as a generic solver for different problems. We showed not just a reformulation of the graph-coloring problem to the k-clique problem, but we also introduced symmetry-breaking techniques in this formulation. In our previous paper, we already showed that preconditioning can be very efficient for such k-partite graphs. With the aid of strong preconditioning and a good clique-solver, our approach proved to be quite efficient in some cases. Note that this special question—the existence of a k-clique in a k-partite graph—has become more important in the last years (see [21–25]).

The present paper is a rather preliminary work, and there are many improvements possible to the present approach. It is clear that preconditioning of the auxiliary graph is crucial for effective solutions (see Table 6), so searching for more-effective preconditioning may lead to much better applications. Further, preconditioning of the original graph may prove its importance as well [26]. In their paper showing results on representative formulation, the authors of [17] expressed the importance of node ordering. It is left unanswered if such an effect would speed up the present approach or not, and if it would, what is the best possible reordering of the nodes. Finally, the k-clique search was performed by a program aimed to work on general graphs, while the present graphs have a special property: they are k-partite. Using a specialized k-clique solver that exploits the k-partite property may also be useful.

Author Contributions: Conceptualization, S.S. and B.Z.; methodology, S.S. and B.Z.; software, S.S. and B.Z.; validation, S.S. and B.Z.; formal analysis, S.S. and B.Z.; investigation, S.S. and B.Z.; resources, S.S. and B.Z.; data curation, S.S. and B.Z.; writing—original draft preparation, S.S. and B.Z.; writing—review and editing, S.S. and B.Z.; visualization, S.S. and B.Z.; supervision, S.S. and B.Z.; project administration, S.S. and B.Z.; funding acquisition, S.S. and B.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported by the National Research, Development and Innovation Office—NKFIH Fund No. SNN-135643.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bron, C.; Kerbosh, J. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM* **1973**, *16*, 575–577. [CrossRef]
2. Carraghan, R.; Pardalos, P.M. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **1990**, *9*, 375–382. [CrossRef]
3. Östergård, P.R.J. A fast algorithm for the maximum clique problem. *Discret. Appl. Math.* **2002**, *120*, 197–207. [CrossRef]
4. Corrádi, K.; Szabó, S. A combinatorial approach for Keller's conjecture. *Period. Math. Hungar.* **1990**, *21*, 91–100. [CrossRef]
5. Szabó, S.; Zaválnij, B. Reducing hyper graph coloring to clique search. *Discret. Appl. Math.* **2019**, *264*, 196–207. [CrossRef]
6. Depolli, M.; Szabó, S.; Zaválnij, B. An Improved Maximum Common Induced Subgraph Solver. *MATCH Commun. Math. Comput. Chem.* **2020**, *84*, 7–28.
7. Szabó, S.; Zaválnij, B. Clique search in graphs of special class and job shop scheduling. *Mathematics* **2022**, *10*, 697. [CrossRef]
8. Cornaz, D.; Jost, V. A one-to-one correspondence between colorings and stable sets. *Oper. Res. Lett.* **2008**, *36*, 673–676. [CrossRef]
9. Szabó, S.; Zaválnij, B. Reducing graph coloring to clique search. *Asia Pac. J. Math.* **2016**, *3*, 64–85.
10. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: New York, NY, USA, 2003.
11. Papadimitriou, C.H. *Computational Complexity*; Addison-Wesley Publishing Company, Inc.: Reading, MA, USA, 1994.
12. de Lima, A.M.; Carmo, R. Exact Algorithms for the Graph Coloring Problem. *Rev. Inf. Teórica Apl.* **2018**, *25*, 57–73. [CrossRef]
13. Campelo, M.; Correa, R.; Frota, Y. Cliques, holes and the vertex coloring polytope. *Inf. Process. Lett.* **2004**, *89*, 159–164. [CrossRef]
14. Szabó, S. Parallel algorithms for finding cliques in a graph. *J. Phys. Conf. Ser.* **2011**, *268*, 012030. [CrossRef]
15. Szabó, S.; Zaválnij, B. A different approach to maximum clique search. In Proceedings of the 2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 20–23 September 2018; pp. 310–316.

16. Zhou, Z.; Li, C.; Huang, C.; Xu, R. An exact algorithm with learning for the graph coloring problem. *Comput. Oper. Res.* **2014**, *51*, 82–301. [CrossRef]
17. Cornaz, D.; Furini, F.; Malaguti, E. Solving vertex coloring problems as maximum weight stable set problems. *Discret. Appl. Math.* **2017**, *217*, 151–162. [CrossRef]
18. Malaguti, E.; Monaci, M.; Toth, P. An exact approach for the vertex coloring problem. *Discret. Optim.* **2011**, *8*, 174–190. [CrossRef]
19. Brélaz, D. New methods to color the vertices of a graph. *Commun. ACM* **1979**, *22*, 251–256. [CrossRef]
20. Culberson, J.C. *Iterated Greedy Graph Coloring and the Difficulty Landscape*; Technical Report; University of Alberta: Edmonton, AB, Canada, 1992.
21. Grünert, T.; Irnich, S.; Zimmermann, H.-J.; Schneider, M.; Wulfhorst, B. Finding all *k*-cliques in *k*-partite graphs, an application in textile engineering. *Comput. Oper. Res.* **2002**, *29*, 13–31. [CrossRef]
22. Kliem, J. A new *k*-partite graph *k*-clique iterator and the optimal colored Tverberg problem for ten colored points. *arXiv* **2022**, arXiv:2112.04268.
23. Mirghorbani, M.; Krokhmal, P. On finding *k*-cliques in *k*-partite graphs. *Optim. Lett.* **2013**, *7*, 1155–1165. [CrossRef]
24. Phillips, C.A.; Wang, K.; Baker, E.J.; Bubier, J.A.; Chesler, E.J.; Langston, M.A. On Finding and Enumerating Maximal and Maximum *k*-Partite Cliques in *k*-Partite Graphs. *Algorithms* **2019**, *12*, 23. [CrossRef]
25. Segundo, P.S.; Furini, F.; León, R. A new branch-and-filter exact algorithm for binary constraint satisfaction problems. *Eur. J. Oper. Res.* **2022**, *299*, 448–467. [CrossRef]
26. Strash, D.; Thompson, L. Effective Data Reduction for the Vertex Clique Cover Problem. In *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. 2022. pp. 41–53. Available online: https://epubs.siam.org/doi/10.1137/1.9781611977042.4 (accessed on 9 July 2022).