Clique search in simultaneously partitioned graphs

1st Peter Schmidt
Institute of Mathematics and Informatics
University of Pecs
Pecs, Hungary
Doctoral School of Applied Informatics and Applied Mathematics
Obuda University
Budapest, Hungary
psmitt@gamma.ttk.pte.hu

2nd Sandor Szabo
Institute of Mathematics and Informatics
University of Pecs
Pecs, Hungary
sszabo7@hotmail.com

3rd Bogdan Zavalnij
Alfréd Rényi Institute of Mathematics
Budapest, Hungary
bogdan@renyi.hu

***Abstract*—Many important applied combinatorial optimization problems can be reduced to locating a k-clique in a graph and consequently, it can handled by a clique solver. In the particular case when the nodes of the graph are well colored with k-colors, that is, when we deal with a k-partite graph, then the clique search algorithm can be speed up.**

In this paper we point out that there are naturally occurring combinatorial optimization problems that can be reduced to k-clique problems but the underlying graph admits more than one k-partitioning. We refere to such a graph as simultaneously k-partitioned graph or b-fold-k-partite graphs. The presence of additional k-partitions of the nodes can improve the efficiency of the clique solver. In addition the k-partitions are playing a part in the preconditioning phase of the problems.

I. Introduction

In this work the graphs we consider have finitely many nodes and finitely many edges. In addition the graphs do not have multiple edges and do not have loops. We refer to such graphs as finite simple graphs. A set of k mutually adjacent nodes in a finite simple graph is called a k-clique. The following problem is known as the k-clique problem.

Problem 1: Given a finite simple graph G and a positive integer k. Decide if G contains a k-clique.

A coloring of the nodes of a given finite simple graph G such that each node receives exactly one color and adjacent nodes never receive the same color is called a proper coloring or well coloring or legal coloring. One may use the numbers $1, 2, \ldots, k$ as colors. The set of nodes of nodes of G receiving color i is called the i-th color class and is denoted by C_i. Note that the colors classes are pair-wise disjoint. The colors class C_i is an independent set of G and the set of nodes of G is a union of the color classes $C_1, \ldots, C_k$. The fact that the nodes of G can be legally colored using k colors can be expressed saying that G is a k-partite graph.

The k-clique problem is computationally demanding since it is an NP-complete problem [12]. On the other hand, the problem plays an important role in discrete applied mathematics. The variant of the k-clique problem in which the finite simple graph is a k-partite graph has many interesting and important applications. This k-partite version of the k-clique problem is still an NP-complete problem – we skip the details, but the most straightforward example is the reduction of a SAT problem, where the graph has so many nodes as we have literals in the SAT problem with k equals to the number of clauses. However, one can exploit the extra property of the graph and the k-clique search in k-partite graph is more efficient than in the general case. (On k-clique search in k-partite graphs see: [10], [14], [16].)

For a finite simple graph G there is an integer k such that G contains a k-clique but G does not contain any $(k+1)$-clique. This well defined integer k is called the clique number of G and denoted by $\omega(G)$. For a finite simple graph G there is an integer k such that G contains an independent set of k elements but G does not contain any independent set of $(k+1)$ element. This uniquely determined integer k is called the stability number of G and denoted by $\alpha(G)$. Obviously, the clique number of a graph is equal to the stability number of the complement of the graph.

A struction is a transformation on a finite simple graph G that produces a graph G' for which the stability number is one less than the stability number of G. The construction introduced by [8]. By the comment we just made about the connection between the clique and stability numbers implies that the struction construction is relevant to the clique number of a graph as well.

In [20] it was shown that the k-partite graph k-clique problem can be reduced to a $(k-1)$-partite graph $(k-1)$-clique problem using struction construction. One of the goal of the present work is to extend this result for finite simple graphs whose nodes can be legally colored with k colors more than one ways, that is, to graphs that admit more than one k partitions.

In this paper we will assume that the finite simple graph G has more than one simultaneous k-partitions. We will show that the presence of more than one k-partitions can be exploited when one is preconditioning the given graph G. (More on preconditioning see: [1], [4], [9], [11], [13], [17], [18].) As an example we will show that auxiliary graphs for Keller's conjecture problem possess such property.

The structure of the paper is as follows. First we describe in details the above mentioned example. Second, we review some

of the simple reduction rules from [20]. Third, we describe the struction transformation in general, and show that the k-partite property can be preserved after such transformation. Fourth, we describe how can we do better preconditioning for graphs with multiple k-partitions. Fifth, we detail possible parallelization methods. Finally, we describe some numerical experiments.

II. Keller's conjecture

We say that a family of translated copies of an n-dimensional unit cube tiles the n-dimensional space if the interiors of any two distinct cubes are disjoint and the union of all the cubes of the family is equal to the whole space. Two n-dimensional cubes sharing a common $(n-1)$-dimensional face is called a twin. In 1930 O. H. Keller conjectured that an n-dimensional cube tiling always has a twin. It turned out that for $n \leq 7$ the appearance of the twins is inevitable but for $n \geq 8$ there are cube tilings without twins.

In order to study Keller's problem we define a graph Γ. The nodes of Γ are the ordered n-tuples $(a_1, \ldots, a_n), 0 \leq a_1 \leq 3, \ldots, 0 \leq a_n \leq 3$. Two distinct nodes $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ are adjacent in Γ if $a_i - b_i$ congruent to 2 modulo 4 for some $i, 1 \leq i \leq n$. If there is exactly one $i, 1 \leq i \leq n$ for which $a_i - b_i$ congruent to 2 modulo 4, then we call the edge a twin edge. As a final step of the construction of Γ we remove the twin edges.

Plainly, Γ has 4^n nodes. We set $k = 2^n$. The connection between Keller's problem and the graph Γ is the following. If Γ contains a k-clique, then there is a counter-example for Keller's conjecture.

To a node $(a_1, \ldots, a_n)$ of Γ we assign a color $(c_1, \ldots, c_n)$. The number c_i is defined by $a_i = 2c_i + d_i$, where $c_i \in \{0, 1\}, d_i \in \{0, 1\}$. (We divided a_i by 2, where c_i is the quotient and d_i is the remainder.) Note that the number of colors we are using is k. Further note that if two distinct nodes of Γ are from the same color class cannot be adjacent. Therefore the graph Γ is a k-partite graph. We are looking for a k-clique in a k-partite graph. (For further details see [5].)

When we divide a_i by 2 we may restrict the remainder d_i into the set $\{0, -1\}$ instead of the set $\{0, 1\}$. In this way we can define a new coloring of the nodes of the graph Γ. As a matter of fact, one can define many colorings of the nodes of Γ changing the set $\{0, 1\}$ to $\{0, -1\}$ independently for each of the n components of the node $(a_1, \ldots, a_n)$. Thus Γ admits k simultaneous k-partitions.

III. Use of multiple colorings

We introduce some simple reduction rules that are crucial in the present work. We just list them and do not detail them, and obviously no proof is given. The reader can find these in [20].

Let G be a finite simple graph whose nodes are legally colored using k-colors and let $C_1, \ldots, C_k$ be the colors classes of this coloring. Further let us suppose that we are looking for a k-clique in G. We describe a situation when a vertex v of G can be deleted from G without loosing any k-clique of G. We say that that the color index of the vertex v is t if v has neighbors in exactly t of the colors classes $C_1, \ldots, C_k$. If the color index of v is less than $k-1$, then v cannot be a node of a k-clique in G and consequently v can be deleted from G. The same can applied for edges as well.

When there are more than one legal colorings of the nodes of G, then we can consider color indices of nodes and edges with respect to various colorings. This increases the chance of deleting nodes and edges.

On the other hand, multiple coloring can enhance the clique search itself. The clique search algorithm [19] branches on the smallest color class in case of k-clique search in a k-partite graph. Naturally, if there are several colorings present one can choose the smallest color class among all of them, thus reducing the size of the search tree.

IV. Struction transformations

We introduce some notations. Let $V = \{1, \ldots, n\}$ be the set of nodes of the graph G. We may assume that node 1 is the pivot node, that is, the node which plays a pivotal role in the struction construction. This is only a matter of renaming the nodes of the graph G.

Let $A = \{a_1, \ldots, a_r\}$ be the set of neighbors of the pivot node 1 and let $B = \{b_1, \ldots, b_s\}$ be the set of non-neighbors of the pivot node 1. Using the set B we construct a new set C. We list $b_1, \ldots, b_s$ is fixed during the reasoning. Note that the argument we follow is valid for any of the possible orderings of the elements of B.

The ordered pair (b_α, b_β) is an element of C whenever the unordered pair $\{b_\alpha, b_\beta\}$ is an edge of G and $\alpha < \beta$. We denote such an ordered pair by $c(b_\alpha, b_\beta)$.

The struction transformation constructs the graph G' from the graph G. The set of nodes of G' is $A \cup C$. Two distinct elements a_i and a_j from A are adjacent in G' if the unordered pair $\{a_i, a_j\}$ is an edge of G. Otherwise, a_i and a_j are not adjacent in G'.

Two distinct elements $c(b_\alpha, b_\beta)$ and $c(b_\gamma, b_\delta)$ from C are adjacent in G' if $b_\alpha = b_\gamma$ and the unordered pair $\{b_\beta, b_\delta\}$ is an edge of G. Otherwise, $c(b_\alpha, b_\beta)$ and $c(b_\gamma, b_\delta)$ are not adjacent in G'.

An element $c(b_\alpha, b_\beta)$ from C and an element a_i from A are adjacent in G' if the unordered pairs $\{b_\alpha, a_i\}$ and $\{b_\beta, a_i\}$ are edges of G. Otherwise, the elements $c(b_\alpha, b_\beta)$ and a_i are not adjacent in G'.

Note that the structure of the graph G' depends of the ordering of the elements of B. Different orderings of the elements of B may lead to non-isomorphic G' graphs. We state here two Lemmas and a Theorem, for proof see Ebenegger at al [8].

Lemma 1: If Δ' is a clique in G', then there is a clique Δ in G, where $|\Delta| = |\Delta'| + 1$

Lemma 2: If Δ is a clique in G, then there is a clique Δ' in G', where $|\Delta'| = |\Delta| - 1$

The following result is a corollary to Lemmas 1 and 2.

Theorem 1: $\omega(G) - 1 = \omega(G')$

A. Node coloring and structions

Let us suppose that the nodes of the graph G are legally colored with the colors $1, \ldots, k$. We will use the function $f : V \to \{1, \ldots, k\}$ to describe the coloring of the nodes. Here the equation $f(v) = i$ means that node v receives color i.

We may assume that $f(1) = 1$ holds, that is, the pivot element is colored with color 1. The reason is the following. If $f(1) = 1$ does not hold, say $f(1) = 2$ holds, then we exchange the roles of the colors 1 and 2.

As we pointed out, the ordering of the elements of B is fixed in the course of the construction. As we do not assume anything about this ordering we cannot assume anything about the ordering of of the assigned colors $f(2), \ldots, f(n)$. Note, that in contrast our previous work [20] does depend on a special ordering of nodes. This extension is crucially needed if one is to use the struction transformation for multiple k-partite graphs in order to preserve that property as it will be stated in Lemma 5.

Theorem 2: Let G be a finite simple graph and let G' be the graph produced by a struction transformation. If the nodes of G can be legally colored using k colors, then the nodes of G' can be legally colored using $k - 1$ colors.

Proof. Using the legal coloring of the nodes of G, that is, using the function f we define a coloring of the nodes of the struction graph G'. We describe the coloring by the function $g : (A \cup C) \to \{1, \ldots, k\}$. We set

$$g(a_i) = f(a_i)$$

for each $a_i \in A$ and we set

$$g(c(b_\alpha, b_\beta)) = \begin{cases} f(b_\beta) & \text{if } f(b_\beta) \neq 1 \\ f(b_\alpha) & \text{if } f(b_\beta) = 1 \end{cases} \tag{1}$$

for each $c(b_\alpha, b_\beta) \in C$. (It will turn out that the range of the function g is in fact $\{2, \ldots, k\}$ and not $\{1, \ldots, k\}$ as we defined here. But this fact will came to light at the end of the proof of the Theorem.)

First we try to establish that the function g provides a legal coloring of the nodes of G'. We pick an edge of G' and try to verify that the end nodes of the edge receive distinct colors.

Let us consider an edge of type $\{a_i, a_j\}$ of G', where $a_i, a_j \in A$. Assume on the contrary that $g(a_i) = g(a_j)$ holds. The computation

$$f(a_i) = g(a_i) = g(a_j) = f(a_j)$$

leads to the contradiction that the end nodes of the edge $\{a_i, a_j\}$ of G receive the same color.

Let us consider an edge of type $\{c(b_\alpha, b_\beta), c(b_\alpha, b_\delta)\}$ of G', where $c(b_\alpha, b_\beta), c(b_\alpha, b_\delta) \in C$. We distinguish the following four cases.

$$\begin{array}{llll} f(b_\beta) = 1, & f(b_\delta) = 1, & f(b_\beta) \neq 1, & f(b_\delta) = 1 \\ f(b_\beta) = 1, & f(b_\delta) \neq 1, & f(b_\beta) \neq 1, & f(b_\delta) \neq 1. \end{array}$$

Case 1: $f(b_\beta) = 1$, $f(b_\delta) = 1$ leads to the contradiction that the end nodes of the edge $\{b_\beta, b_\delta\}$ of G receive the same color, namely color 1. The reader will notice that the unordered pair $\{b_\beta, b_\delta\}$ is an edge of the graph G. This follows from the definition of $\{c(b_\alpha, b_\beta), c(b_\alpha, b_\delta)\}$.

For the remaining cases assume on the contrary that $g(c(b_\alpha, b_\beta)) = g(c(b_\alpha, b_\delta))$ holds.

Case 2: $f(b_\beta) \neq 1$, $f(b_\delta) = 1$ means that

$$\begin{aligned} f(b_\alpha) &= g(c(b_\alpha, b_\delta)) \\ &= g(c(b_\alpha, b_\beta)) \\ &= f(b_\beta) \end{aligned}$$

which leads to the contradiction that the end nodes of the edge $\{b_\alpha, b_\beta\}$ of G receive the same color.

Case 3: $f(b_\beta) = 1$, $f(b_\delta) \neq 1$ means that

$$\begin{aligned} f(b_\alpha) &= g(c(b_\alpha, b_\beta)) \\ &= g(c(b_\alpha, b_\delta)) \\ &= f(b_\delta) \end{aligned}$$

which leads to the contradiction that the end nodes of the edge $\{b_\alpha, b_\delta\}$ of G receive the same color.

Case 4: $f(b_\beta) \neq 1$, $f(b_\delta) \neq 1$. The computation

$$\begin{aligned} f(b_\beta) &= g(c(b_\alpha, b_\beta)) \\ &= g(c(b_\alpha, b_\delta)) \\ &= f(b_\delta) \end{aligned}$$

leads to the contradiction that the end nodes of the edge $\{b_\beta, b_\delta\}$ of G receive the same color. A moment contemplation will convince the reader that the unordered pair $\{b_\beta, b_\delta\}$ is an edge of G.

Let us consider an edge of type $\{c(b_\alpha, b_\beta), a_i\}$ of G', where $c(b_\alpha, b_\beta) \in C$ and $a_i \in A$. Assume on the contrary that $g(c(b_\alpha, b_\beta)) = g(a_i)$ holds. We distinguish two cases depending on either $f(b_\beta) \neq 1$ or $f(b_\beta) = 1$.

Case 1: $f(b_\beta) \neq 1$. The computation

$$\begin{aligned} f(b_\beta) &= g(c(b_\alpha, b_\beta)) \\ &= g(a_i) \\ &= f(a_i) \end{aligned}$$

leads to the contradiction that the end nodes of the edge $\{b_\beta, a_i\}$ of G receive the same color.

Case 2: $f(b_\beta) = 1$. The computation

$$\begin{aligned} f(b_\alpha) &= g(c(b_\alpha, b_\beta)) \\ &= g(a_i) \\ &= f(a_i) \end{aligned}$$

leads to the contradiction that the end nodes of the edge $\{b_\alpha, a_i\}$ of G receive the same color.

Next we try to establish that the function g does not take the value 1 on. In other words, we try to show that the nodes of the struction graph G' are colored with the colors $2, \ldots, k$. We pick a node of G' and try to show that the node is not colored with color 1.

Let us consider a node of type a_i, where $a_i \in A$ and assume on the contrary that $g(a_i) = 1$. Note that the unordered pair

$\{1, a_i\}$ is an edge of G. (Remember the neighbors of the pivot node 1 are the elements of A.) Using $f(a_i) = g(a_i) = 1$ we end up with the contradiction that the end nodes of edge $\{1, a_i\}$ of G receive the same color.

Let us consider a node of type $c(b_\alpha, b_\beta)$ of G', where $c(b_\alpha, b_\beta) \in C$. We distinguish two cases depending on either $f(b_\beta) \neq 1$ or $f(b_\beta) = 1$.

Case 1: $f(b_\beta) \neq 1$. As $g(c(b_\alpha, b_\beta)) = f(b_\beta)$ the right hand side of the equation is different from 1.

Case 2: $f(b_\beta) = 1$. The definition of the function g gives that $g(c(b_\alpha, b_\beta)) = f(b_\alpha)$. Let us assume on the contrary that $f(b_\alpha) = 1$. Note that the unordered pair $\{b_\alpha, b_\beta\}$ is an edge of G. Thus, the end nodes of this edge of G receive the same color.

□

V. Reduction of set C

Let G be a finite simple graph and let G' be the graph we get from G applying the struction transformation. As stated previously, the set of nodes of G' is $A \cup C$.

In [20] the authors adopted the struction operation to a k-partite graph G to construct a G^* graph, such as the new graph admits a $(k-1)$-clique if and only if the same is true for the original one. As it turned out G^* is a $(k-1)$-partite graph and the equation $\omega(G) - 1 = \omega(G^*)$ holds. Exploiting the k-partite property of G the set C^* which plays an analogous role to the set C in the generic construction, typically has fewer elements than C. As a consequence in case of a k-partite graph the constructed graph G^* has fewer elements compared with the graph G' resulted by the generic struction operation. In this section we continue this line of investigation. We will consider finite simple graphs that admit several simultaneous k-partitions.

A. Definitions

Let G be a finite simple graph whose nodes are legally colored using s colors. Suppose that $CC_1, \ldots, CC_s$ are the color classes of this coloring. We introduce a color index of a node v of G. The color index of v is the number of color classes CC_i that contain at least one neighbor of the node v. Since there may be many ways for legally coloring the nodes of the graph G we cannot talk about a definite color index of a node. The colors index is always defined with respect to a fixed legal coloring of the nodes of the graph.

Next we assume that G is a finite simple k-partite graph. We quote two lemmas from [20] without proofs. These results were used to preprocess large graph before we start a large scale clique search procedure. In this section we will extend this preconditioning technique for the case when the nodes of the given graphs are colored more than once.

Lemma 3: [20] If the colors index of v of the the graph G is at most $k-2$, then v can be deleted from G without losing any k-clique.

Lemma 4: [20] Let D be the color class of the pivot node p in the graph G. Let C^* be the set of the elements in form $c(b_\alpha, b_\beta)$, where the unordered pair $\{b_\alpha, b_\beta\}$ is an edge of G and the $b_\alpha < b_\beta$, $b_\alpha \in D$ conditions hold. In the struction construction we may replace the set C by the (typically) smaller set C^*.

Let $G = (V, E)$ be a finite simple graph and let b, k be positive integers. We set $CO = \{1, \ldots, k\}$. To each node v of G we assign an ordered b-tuple $[co_1, \ldots, co_b]$, where $co_i \in CO$. The intuitive meaning of this assignment is that we assign the colors $co_1, \ldots, co_b$ to the node v of G. Here the numbers $1, \ldots, k$ are used as colors. We may use the functions

$$f^1 : V \to CO, \ldots, f^b : V \to CO$$

to describe these b simultaneous colorings of the nodes of G. The equation $f^i(v) = co_i$ means that node v receives color co_i as the i^{th} color. The b simultaneous colorings $f^1, \ldots, f^b$ together are called a simultaneous legal coloring if $f^i(u) \neq f^i(v)$ whenever the unordered pair $\{u, v\}$ is an edge of the graph G for each i, $1 \leq i \leq b$. We refer to the function f^i as the i^{th} coloring of the nodes of G. The set $CC^i_j = \{v : f^i(v) = j, \ \ v \in V\}$ will be called the j^{th} color class of the i^{th} coloring f^i. The sets $CC^i_1, \ldots, CC^i_k$ form a partition of the set V. The nodes of the graph G has b simultaneous partitions as i ranges through the values $1, \ldots, b$. We will call the graph G a b-fold-k-partite graph. The $b = 1$ particular case is the k-partite graph described in [20].

We may assume that $f^1(1) = 1$ holds, that is, the pivot element receives color 1 in the 1^{st} coloring because this is only a matter of exchanging the colors in the 1^{st} coloring. Similarly, we may assume that $f^2(1) = 1$ holds, that is, the pivot element receives color 1 in the 2^{nd} coloring. finally, we may assume that $f^b(1) = 1$ holds, that is, the pivot element receives color 1 in the b^{th} coloring.

A b-fold-k-partite graph admits b simultaneous legal colorings using k colors. Therefore, in this particular case Lemma 3 can be applied b times. This increases the chance of deleting nodes from the tested graph.

One may be tempted to apply Lemma 4 b times simultaneously for each of the b legal k-colorings of the nodes of a b-fold-k-partite graph. However, an inspection of the proof of Lemma 4 reveals that Lemma 4 is indeed applicable to the 1^{st} coloring of the graph but it is not necessarily applicable to the 2^{nd} coloring as the argument requires rearranging of the elements of the set B and we are not in liberty to rearrange the elements any longer. (Namely, this might undo the rearrangement used in connection with the 1^{st} coloring.)

In order to make our presentation clearer we emphasize this point by reiterating that if the nodes of the graph G we work with admit b simultaneous colorings each of which uses k colors, then the nodes of the result of the struction construction G' admit b simultaneous colorings each of which uses $(k-1)$ colors. The proof of this proposition cannot be based on the publishes proof when the nodes of G has only one coloring using k colors. In other words we cannot use repeteadly the proof of the $b = 1$ particular case.

B. Reducing C for b-fold-k-partite graphs

In this section we present an improvement of Lemma 4 for b-fold-k-partite graphs. Let $G = (V, E)$ be a finite simple b-fold-k-partite graph and suppose that the legal k-colorings of the nodes of G are given by the functions $f^1, \ldots, f^b$. We suppose that the nodes of G are $1, \ldots, n$ and the node 1 is the pivot node of the struction transformation. As we have seen earlier we may assume that $f^1(1) = 1, \ldots, f^b(1) = 1$. In plain English we may assume that the pivot node 1 receives color 1 at each of the b simultaneous colorings of the nodes of G.

In the proof of Lemma 4 the nodes of G are listed such that $v < u$ implies $f^1(v) < f^1(u)$. Using this setting we carry out a struction construction and from the graph G we get a new graph G'. Using the colorings $f^1, \ldots, f^b$ of the nodes of the graph G we construct colorings $g^1, \ldots, g^b$ of the nodes of the graph G'. We set

$$g^i(a_j) = f^i(a_j)$$

for each i, $1 \leq i \leq b$ and $a_j \in A$ and we set

$$g^i(c(b_\alpha, b_\beta)) = \begin{cases} f^i(b_\beta) & \text{if } f^i(b_\beta) \neq f^i(1) = 1 \\ f^i(b_\alpha) & \text{if } f^i(b_\beta) = f^i(1) = 1 \end{cases} \quad (2)$$

for each i, $1 \leq i \leq b$ and $c(b_\alpha, b_\beta) \in C$.

The next result is a consequence of Theorem 2.

Lemma 5: The resulting graph G' is a b-fold-$(k-1)$-partite graph.

Next we inspect the set C and delete from it nodes according to Lemmas 3 and 4. According to Lemma 4, that is, using the color index deletion with respect to the 1st coloring all the nodes $c(b_\alpha, b_\beta)$ where $f^1(b_\alpha) \neq 1$ can be deleted from C.

We claim, that using the colorings $f^2, \ldots, f^b$ some of the nodes $c(b_\alpha, b_\beta)$ can be deleted from C, where $f^i(b_\alpha) \neq 1, f^i(b_\beta) \neq 1$ for some i, $2 \leq i \leq b$.

Let us consider the node $c(b_\alpha, b_\beta)$. The color of this node at the coloring f^i is $g^i(c(b_\alpha, b_\beta)) = f^i(b_\beta)$ and it may have no neighbors in the color class of the color $f^i(b_\alpha)$. This node can be deleted, if there does not exist a node $b_\delta \in B$ of G, such that the nodes b_α, b_β, b_δ form a triangle (3-clique) in G, and $b_\delta > b_\alpha$, $f^i(b_\delta) = 1$. If such node exists, then the equations

$$g^i(c(b_\alpha, b_\beta)) = f^i(b_\beta), \quad g^i(c(b_\alpha, b_\delta)) = f^i(b_\alpha)$$

hold. As the nodes $c(b_\alpha, b_\beta)$ and $c(b_\alpha, b_\delta)$ are connected by an edge, then there is a neighbor of $c(b_\alpha, b_\beta)$ in the color class of the color $f^i(b_\alpha)$, and we do not delete it.

In the alternative case, when there is no such a node b_δ the color class of the color $f^i(b_\alpha)$ does not contain any neighbor of $c(b_\alpha, b_\beta)$ in the color class of the color $f^i(b_\alpha)$. Now the color index of the node $c(b_\alpha, b_\beta)$ with respect to the coloring f^i, is too small. Consequently, the node $c(b_\alpha, b_\beta)$ can be deleted.

The essential point of the argument above is the following. The only obstruction of applying the color index based deletion argument to the node $c(b_\alpha, b_\beta)$ of the graph G', in connection with the color class of the color $f^i(b_\alpha)$, is the existence of a 3-clique b_α, b_β, $b_\delta, b_\delta > b_\alpha$ in the graph G. Therefore, the non-existence of such 3-cliques is a certificate that the color index based deletion argument applicable in connection with the color class of color $f^i(b_\alpha)$. One should inspect all color classes of the color $f^i(b_\alpha)$ for each $i, 2 \leq i \leq b$ of the graph G' and in each case one has to carry out an exhaustive search to locate a possible obstructing 3-clique. If a color class is located without obstructing 3-clique, then the color index based deletion argument can be applied.

One would like to use this method iteratively in the sense that if a node is deleted from the graph G', then we start to check whether the new graph has deletable node.

VI. Parallelizing clique search problem into smaller instances

The parallelization of a problem, that is dividing the problem into smaller problem instances, is a well known approach for hard problems to be solved on a parallel machine. We describe three methods here.

First, there is the well known approach of dividing the problem by the first levels of the search tree. The usual drawback is that as we use more and more levels, the subproblems become unbalanced [15].

So we propose two other methods, based on the struction transformation. These may (or may not) produce more balanced division (in some cases.)

Lemma 6: The subgraph of G' induced by C^* has at least as many separate components as many nodes $c(b_\alpha, b_\beta)$ exist where $f^1(b_\alpha) = 1$.

Proof. As C^* has only nodes $c(b_\alpha, b_\beta)$ for which $f^1(b_\alpha) = 1$ and due to the struction rule $c(b_\alpha, b_\beta)$ and $c(b_\gamma, b_\delta)$ cannot be connected by an edge if $b_\alpha \neq b_\gamma$. Therefore, each node b_α determines a separate component of the subgraph, composed of nodes $c(b_\alpha, x)$, which we can designate as C^*_α. Due to the struction rule, there are no edges between the components C^*_α and C^*_γ, if $b_\alpha \neq b_\gamma$. □

Lemma 7: If the nodes of a Δ clique contained by $A \cup C^*$, then there is a $A \cup C^*_\alpha$ for some α, so that it contains all the nodes of Δ.

Proof. The nodes of Δ either contained fully in A, or some nodes $\{x_1, x_2, \ldots, x_d\} \subset C^*$. These nodes cannot be in the node set of different components such as C^*_α and C^*_γ at the same time, as there is no edges between the graphs induced by these sets. Thus, there is one (or more) G'_α subgraph containing fully Δ. □

Consequently, the search for a k-clique in G' can be divided into sub-searches within the G'_α sub-graphs, this is the second proposed method.

Alas, the set A is present in all of the sub-problems, making them overlapping from algorithmic viewpoint. To mend this we can divide the problem the following way as the third method. The zeroth problem will be the graph induced by A. For each of the following problems we choose a node c_i from

the set C^* and use its neighborhood. The sub-problem is the graph induced by $N(c_i) \cap (A \cup C^* \setminus \{c_1, c_2, \ldots, c_{i-1}\})$.

VII. Numerical experiments for the Keller's conjecture

The aim of the present paper was to show that the multiple coloring of the auxiliary graph for the Keller's conjecture can help to solve the problem. We aimed at the most hard case, the keller-7 graph, but due to different reasons described below we did not succeeded. So we present here negative results and propose some explanation.

As described in Section II the numerical problem of the Keller's conjecture can be reduced to a suitable auxiliary graph Γ with 4^n nodes, where a k-clique, $k = 2^n$ would be a counterexample. More precisely the graphs can be reduced in such a way, that the first node is forced into the solution, that is the new graph is the neighborhood of the first node. These graphs among others used for maximum clique solvers for evaluation and solved up to $n = 5$ by the best solvers. By specialized approach, such as among others graph automorphisms – and not by a general solver – the problems for $n = 6$ was solved by Peter Shor, and for case $n = 7$ were solved by Debroni at al [6] proving that $\omega(\Gamma), n = 7$ is equal to 124 (123 for the reduced graph). The calculations were performed on a cluster of computers and took years of CPU time.

Here we constrain ourselves to simpler problem, namely the one which is asked by the Keller's conjecture itself, we would just decide if the reduced graph has a k-clique of size $k = 2^n - 1$ or not. For the case $n = 7$ the question if there is a 127-clique in the graph. Debroni at al details this question in their work, and solve this problem on a 64 CPU computer using 40 days of CPU time. Our approach here is to use a general solver. We tried to solve it directly by a k-clique solver we used in [19], but as expected it did not finish in 24 hours. (By checking the program output, our guess that it will take more than 10 years to finish.)

During our numerical experiments we tried to used the approach described in the present work and use all 2^n k-colorings for solution, but failed to reach a good result. In the following we describe some notions that we think is behind the negative results.

Note that the graph in question is extremely symmetrical, meaning that the size distribution of color classes are very similar through different colorings. Moreover, if we force some nodes into a clique and take their common neighborhood, this property remains for quite a number of nodes (at least 4-5). Note, that Debroni at al [6] also report that the sub-problems they produced were extremely balanced in therms of computational time.

First, we tried to use multiple coloring inside the clique search for branching rule as well as for a simple node deletion procedure during the search based on color indices. This did not helped much, and having so much colorings at once made the program even slower. So we restricted the number of colorings to choose from and the node deleting by depht level as well as number of colorings to check for color indices. Still the best for keller-6 was 33 seconds vs. 37 seconds by the non specific program.

Second, we divided the problem into subproblems by search level. On second level this produced 1 953 sub-problems, on third level 29 265. The later sub-problems can be solved in one to two hour time, adding up 5-10 years of full CPU time, similar as for a sequential program – that is reasonable for a supercomputer, but obviously has little novelty so we did not do it. The second level division produced similar results. Note, that the sub-problems turned out to be very balanced, which is a good case opposed to other problems.

Third, we tried to use our slightly modified preconditioning program described in [20], [21] for using multiple colorings. The preconditioner also uses the struction transformation but limit it usage that it can blow up the graph only slightly. (See more about reductions that increase the graph size in [9].) It failed spectacularly. Being extremely symmetrical the preconditioner could not delete any nodes or edges.

Fourth, we tried the preconditioner on the sub-divided 1 953 problems described previously (level two), where the running time of the clique search program was a few days. We expected that fixing 3 nodes (one for the general problem described in the beginning of this section, two for the second level) would result in a less symmetrical problem, resulting the preconditioner reducing the problem in an hour time to a smaller one that can be solved again in an hour or so. But the preconditioner was still powerless, although it could delete a few nodes and some edges, these numbers did not reached one percent of the graph – running for days.

Fifth, we tried the second division using struction described above. That resulted in almost 60 thousand subproblems that needed several hours to solve – resulting in worse sum time as the division by levels of the search tree.

Sixth, we blew up the graph using struction transformation. The original graph has 14k nodes, with one struction it grew to 68k, with two structions to 122k nodes. We tried some of the above approaches on these, but all of them was worse than previously.

Seventh and last, using all empirical information collected during these tries, we came up with a pinpointed plan. We planed to divide the problem on the first level of search tree (resulting 63 sub-problems), perform a struction transformation on these (that increases the $\sim$10k node size to $\sim$30k node size graph), to perform an extended preconditioning on these, and hoped that then the reduced graphs can be solved in less time by the clique search program. But again, this did not worked, as the reduction could not reduce the graph in days even by 1 percent.

VIII. Summary

We showed that such multiple k-partitions can be exploited for preconditioning the k-clique search problem. The main result of our work is that our previous methods using color indices [20] and struction transformation by Ebenegger at al [8] can be extended for multiple k-partitite graphs, as well

that the clique search algorithm can use it to reduce the size of the search tree.

The chosen problem of keller-7 turned out too symmetrical to these approaches, so we could collect only negative results which we described in details. We hope, that for less symmetrical problems this approach has still some potential.

REFERENCES

[1] Akiba T., Iwata, Y. Branch-and-reduce exponential/fpt algorithms in practice: A case study of vertex cover. *Theoretical Computer Science.* 609 pp. 211–225, 2016.

[2] Alexe G., Hammer P.L., Lozin V.V., Werra D. de, Struction revisited. *Discrete Applied Mathematics.* **132** (1–3): 27–46, 2003.

[3] Brakensiek, J., Heule, M., Mackey, J. and Narváez, D. *The resolution of Keller's conjecture.* in Peltier, N., Sofronie-Stokkermans, V. (eds.), Automated Reasoning: 10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part I, Lecture Notes in Computer Science, vol. 12166, Springer, pp. 48–65. 2020.

[4] Chang, L., Li, W., Zhang, W. Computing a near-maximum independent set in linear time by reducing-peeling. In: Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17), 1181–1196, 2017.

[5] Corrádi, K., Szabó, S. A combinatorial approach for Keller's conjecture. *Period Math Hung.* 21, 95–100 (1990).

[6] Debroni, J., Eblen, J.D., Langston, M.A., Shor, P., Myrvold, W. and Weerapurage, D. A complete resolution of the Keller maximum clique problem. *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms.* pp. 129–135, 2011.

[7] Depolli, M., Szabó S. and Zaválnij B. An Improved Maximum Common Induced Subgraph Solver. *MATCH Commun. Math. Comput. Chem.* 84 (2020) pp. 7–28.

[8] Ebenegger, Ch., Hammer, P.L., Werra, D. de, Pseudo-Boolean Functions and Stability of Graphs. *North-Holland Mathematics Studies.* Volume 95, pp 83–97, 1984.

[9] Gellner, A., Lamm, S., Schulz, Ch., Strash, D., Zaválnij B.: Boosting Data Reduction for the Maximum Weight Independent Set Problem Using Increasing Transformations. In: *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX).* Society for Industrial and Applied Mathematics (2021) pp. 128–142.

[10] Grünert, T., Irnich, S., Zimmermann, H., Schneider, S.M., Wulfhorst, B. Finding all k-cliques in k-partite graphs, an application in textile engineering. *Comput. Oper. Res.* 2001, 29, 13–31.

[11] Hespe, D., Schulz, C., Strash, D. Scalable kernelization for maximum independent sets. In: 2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX) 223–237. SIAM, 2018.

[12] Karp, R.M. Reducibility Among Combinatorial Problems. In: Complexity of Computer Computations. Eds: R. E. Miller and J. W. Thatcher. New York. Plenum. pp. 85–103.

[13] Lamm S., Sanders P., Schulz C., Strash, D., Werneck R.F., Finding Near-Optimal Independent Sets at Scale. In: Proceedings of the 16th Meeting on Algorithm Engineering and Experimentation (ALENEX'16)

[14] Mirghorbani, M., Krokhmal, P. On finding k-cliques in k-partite graphs. *Optimization Letters.* 7, 1155–1165. 2013.

[15] McCreesh, C. and Prosser, P. The Shape of the Search Tree for the Maximum Clique Problem, and the Implications for Parallel Branch and Bound. *ACM Transactions on Parallel Computing.* Volume 2 Issue 1, May 2015. Article No. 8.

[16] San Segundo, P., Furini, F., León, R. A new branch-and-filter exact algorithm for binary constraint satisfaction problems. *European Journal of Operational Research.* Volume 299, Issue 2, 2022, pp. 448–467,

[17] Strash, D. On the power of simple reductions for the maximum independent set problem. In: T. N. Dinh and M. T. Thai, editors, Computing and Combinatorics (COCOON'16), volume 9797 of LNCS, pages 345–356. 2016.

[18] Szabó, S. Parallel algorithms for finding cliques in a graph, *Journal of Physics: Conference Series.* **268** 012030 DOI:10.1088/1742-6596/268/1/012030

[19] Szabó, S., Zavalnij, B. Combining algorithms for vertex cover and clique search. In: Proceedings of the 22nd International Multiconference INFORMATION SOCIETY – IS 2019, Volume I : Middle-European Conference on Applied Theoretical Computer Science Ljubljana, Slovenia pp. 71–74. 2019.

[20] Szabó S., and B. Zaválnij, Clique search in graphs of special class and job shop scheduling. *Mathematics.* 2022, 10(5), 697.

[21] Szabo, S. and Zavalnij, B. Graph Coloring via Clique Search with Symmetry Breaking. SYMMETRY (BASEL) 14 : 8 Paper: 1574 , 16 p. (2022)

[22] Tarry, G. Le probleme des 36 officiers, C. R. Acad. Franc. pour. l'Avanc. de Sci. Naturel, (1900) **1**, 122–123, (1900) **2**, 170–203.

[23] Weisstein, Eric W. "Mycielski Graph." From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/MycielskiGraph.html