Discrete Applied Mathematics 242 (2018) 118–129

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Decomposing clique search problems into smaller instances based on node and edge colorings

Sándor Szabó, Bogdan Zavalnij *

Institute of Mathematics and Informatics, University of Pecs, H-7624, Pecs, Ifjusag utja 6, Hungary

A R T I C L E I N F O

Article history:
Received 30 November 2016
Received in revised form 17 November 2017
Accepted 8 January 2018
Available online 2 February 2018

Keywords:
k-clique
Maximum clique
Clique search algorithm
Independent set
Branch and Bound
Node coloring
Edge Coloring
Greedy coloring
Combinatorial optimization

A B S T R A C T

To carry out a clique search in a given graph in a parallel fashion, one divides the problem into a very large number of smaller instances. To sort out as many resulted smaller problems as possible, one can rely on upper estimates of the clique sizes. Legal coloring of the nodes of the graphs is a commonly used tool to establish upper bound of the clique size. We will point out that coloring of the nodes can also be used to divide the clique search problem into smaller ones. We will introduce a non-conventional coloring of the edges of the given graph. We will gather theoretical and computational evidence that the proposed edge coloring provides better estimates for the clique size than the node coloring and can be used to divide the original problem into subproblems.

1. Introduction

Let $G = (V, E)$ be a finite simple graph. It means that G has finitely many nodes, and G does not have double edges or loops. Let k be a fixed positive integer. A subgraph Δ of G is called a k-clique if each two distinct nodes of Δ are adjacent, and Δ has k nodes. Sometimes we call a k-clique a clique of size k. A k-clique Δ in G is called a maximal clique if Δ is not a subgraph of any $(k+1)$-clique in G. A k-clique Δ in G is called a maximum clique if G does not contain any $(k+1)$-clique. The size of a maximum clique in G is called as the clique number of G, and it is denoted by $\omega(G)$.

We describe the clique search problems relevant to this paper.

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

Problem 1 is a decision problem, and it is well-known that it belongs to the NP-complete complexity class [9].

Problem 2. Given a finite simple graph G and a positive integer k. List all k-cliques in G.

It is clear that Problem 2 is not a decision problem and that it cannot be computationally less demanding than Problem 1. In other words Problem 2 belongs to the NP-hard complexity class.

Clique search problems have many practical applications, and there is a considerable amount of research devoted to them. For details, see for example [3,8,10,11,13,16,18].

* Corresponding author.
E-mail addresses: sszabo7@hotmail.com (S. Szabó), bogdan@ttk.pte.hu (B. Zavalnij).

https://doi.org/10.1016/j.dam.2018.01.006

Definition 1. We color the nodes of a given finite simple graph G with k colors such that

(1) each node receives exactly one of the colors.
(2) adjacent nodes never receive the same color.

This is the most commonly encountered coloring of the nodes of a graph. We will refer to it as a legal coloring of the nodes of G. If the nodes of G have a legal coloring with k colors, then $\omega(G) \leq k$.

We will point out that coloring of the nodes of the graph helps in dividing the problem into a large number of smaller problems that can be executed in a completely independent fashion. In other words coloring the nodes is relevant in constructing parallel clique search algorithms. This is one of the main results of the paper. As the other main result of this paper, we propose a non-conventional coloring of the edges of a graph, in contrast to edge coloring proposed by Vizing [20].

Definition 2. We color the edges of a graph G with k colors in the following way.

(1) Each edge receives exactly one color.
(2) If x, y, z are distinct nodes of a 3-clique in G, then the edges $\{x, y\}$, $\{y, z\}$, $\{x, z\}$ cannot receive the same color.
(3) If x, y, u, v are distinct nodes of a 4-clique in G, then the edges $\{x, y\}$, $\{u, v\}$ cannot receive the same color.

We call this type of coloring of the edges of G a legal edge coloring. It turns out that if the edges of G can be legally colored with t colors and k is the largest integer for which $k(k-1)/2 \leq t$ holds, then $\omega(G) \leq k$. Therefore, legal coloring the edges of a graph G can be used to establish an upper bound for $\omega(G)$.

The outline of the paper is the following. In Section 2 we will define a set of nodes, the so-called k-clique covering set, that plays a role in dividing a clique search problem into smaller problems. We will point out that legal coloring of the nodes can be used to construct such sets.

In order to use legal edge coloring to divide a clique search problem into smaller problems, we define a k-clique covering edge set in Section 3. We will introduce the concept of quasi-coloring of the nodes and show how this concept can be utilized in constructing k-clique covering edge sets.

We will introduce the concept of derived graph and show that legal edge coloring cannot give weaker clique estimates than the legal node coloring.

In order to compare the performances of greedy node and edge colorings in Section 4, we carried out a large-scale numerical experiment. The benchmark problems and the results are presented in the last part of the paper. The results confirm that the proposed edge coloring method systematically gives us better estimates than legal coloring. We also present examples of when the edge coloring can actually give better bound than the chromatic number.

2. *k*-clique covering node set and coloring of the nodes

Definition 3. Let $G = (V, E)$ be a finite simple graph, and let k be a positive integer. Let $W \subseteq V$. If each k-clique in G has at least one node in W, then we call W a k-clique covering node set of G. (See Fig. 1.)

Let W be a k-clique covering node set in G and let

$$\{v_1, v_2, \ldots, v_n\}$$

be all the nodes in W. Consider the subgraph H_i of G induced by the set of nodes $N(v_i)$ in G for each i, $1 \leq i \leq n$. Here $N(v_i)$ denotes the set of all the nodes of V that are neighbors to v_i.

Let Δ be a k-clique in G. The definition of W states that v_i is a node of Δ for some i, $1 \leq i \leq n$. Consequently, the subgraph H_i contains exactly $k-1$ nodes of the clique Δ. This observation has a clear intuitive meaning. The problem of locating a k-clique in G can be reduced to a list of smaller problems of locating a $(k-1)$-clique in the graph H_i for each i, $1 \leq i \leq n$.

The smaller is the n, the fewer are the subproblems we end up with.

Problem 3. Given a finite simple graph G and a positive integer k, find a minimum size k-clique covering set in G.

Problem 3 cannot be computationally easier than Problem 1, and so Problem 3 belongs to the NP-hard complexity class.

The message is that determining the optimal size of the k-clique covering node sets is a computationally demanding problem. For this reason, instead of working with optimal size k-clique covering node sets, we will work with not necessarily optimal size k-clique covering node sets. There are many widely-used methods in the literature for colorings of the nodes, as for example in [4,6]. One should color the nodes of the graph legally, and then choose the $(k-1)$ biggest color classes $C_1, C_2, \ldots, C_{k-1}$. Let the set of nodes U be the union of these color classes. Clearly, the maximum clique in the subgraph induced by U at most $(k-1)$ as the coloring puts an upper bound on the clique size. From this, it follows that for any k-clique, there should be at least one node outside this set U, so the nodes outside these color classes, namely the nodes in the set $W = V \setminus U$, are forming a k-clique covering node set. This implies that the above described subproblems of searching $(k-1)$-cliques in the H_i induced subgraphs will form a branching in a Branch-and-Bound algorithm. In fact this method was described with some minor modifications in considerable details in [1]. We included it as an example to illustrate our more general approach.