Mathematica Pannonica *New Series* 27 /NS 1/ (2021) 1, 32-47
DOI: 10.1556/314.2020.00005

AKADÉMIAI KIADÓ

SOME ZERO-ONE LINEAR PROGRAMMING REFORMULATIONS FOR THE MAXIMUM CLIQUE PROBLEM

Ákos Beke[1,a], Sándor Szabó[1,*] and Bogdán Zavalnij[1,b]

[1] Institute of Mathematics and Informatics, University of Pécs, Ifjúság u. 6, 7624 Pécs, Hungary
[a] akos85@gmail.com
[b] bogdan@ttk.pte.hu

Communicated by László Tóth

Original Research Paper

Received: Sep 20, 2019 • Accepted: Nov 23, 2020
First published online: Mar 24, 2021

ABSTRACT

Many combinatorial optimization problems can be expressed in terms of zero-one linear programs. For the maximum clique problem the so-called edge reformulation is applied most commonly. Two less frequently used LP equivalents are the independent set and edge covering set reformulations. The number of the constraints (as a function of the number of vertices of the ground graph) is asymptotically quadratic in the edge and the edge covering set LP reformulations and it is exponential in the independent set reformulation, respectively. F. D. Croce and R. Tadei proposed an approach in which the number of the constraints is equal to the number of the vertices. In this paper we are looking for possible tighter variants of these linear programs.

KEYWORDS

combinatorial optimization, maximum cliques,zero-one linear programming, greedy coloring,practical solutions of NP complete problems

MATHEMATICS SUBJECT CLASSIFICATION (2020)

Primary 94B60; Secondary 05B45, 52C22

1. INTRODUCTION

Let $G = (V, E)$ be a finite simple graph. In other words we assume that the vertex set V of G is finite. Further we assume that G does not have loops or double edges. When we deal with such a restricted concept of graphs we may suppose that the set of edges E of G is set of two element subsets of V.

Let U be a subset of V. If each two distinct vertices in U are always adjacent in G, then we say that the subset U induces a clique Δ in G. If U has k elements, then we say that Δ is a clique of size k or simply we say that Δ is a k-clique in G. A k-clique Δ in G is called a maximum clique if G does not contain any clique with size larger than 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. The following two problems are known as the k-clique problem and the maximum clique problem, respectively.

* Corresponding author. E-mail: sszabo7@hotmail.com

PROBLEM 1.1. Given a finite simple graph G and given a positive integer k. Decide if G has a clique of size k.

PROBLEM 1.2. Given a finite simple graph G. Determine the size of the maximum cliques in G.

From the complexity theory of computations we know that these problems are computationally hard. Namely, Problem 1.1, as a decision problem, belongs to the NP complete complexity class. (For details see [4] or [10].) On the other hand there are important practical problems which lead to a k-clique or a maximum clique problem. Many applications are described in [1] and many bench mark problems are given in [5].

The work horses in the majority of the real life clique search computations are the Carraghan-Pardalos [2] and the Östergård [9] algorithms. In their simplest forms these are nothing more than implicit enumeration schemes. Equipped with pruning methods coming from elementary combinatorial considerations such as coloring and matching, well tuned implementations of the algorithms are capable of handling highly non-trivial instances. (For well implemented algorithms see for example [7], [8], [11].)

Our main motivation of studying the 0-1 LP equivalents of the maximum clique problem is to see if the upper estimates provided by the LP machinery can be exploited to improve the performance of the clique search procedures.

2. THE 0-1 LP REFORMULATIONS

In this section we present four 0-1 linear programming equivalents for the maximum clique problem. The first one, the so-called edge reformulation, is the most well-known. In this reformulation the number of variables is equal to n while the number of the constraints is $O(n^2)$, where n is the number of the nodes of the input graph. The edge covering set approach is the next most commonly encountered equivalent. In this case the number of variables is $O(n^2)$ and the number of constraint is n. We will point out that the two reformulations are intimately connected.

F. D. Croce and R. Tadei [3] proposed a formulation in which both the number of variables and the number of constraints are n. The last two reformulations in this section are related to the reformulation of Croce and Tadei. We are looking for variants that possibly are tighter versions of the associated linear programs. A large scale numerical experiment indicates a clear winner among the proposed tightened variants.

Let us consider the following two linear programs.

$$\begin{array}{rcl} \mathbf{Ax} & \leq & \mathbf{b} \\ \mathbf{cx} & \longrightarrow & \max \\ \mathbf{x} & \geq & \mathbf{0} \end{array} \qquad\qquad \begin{array}{rcl} \mathbf{yA} & \geq & \mathbf{c} \\ \mathbf{yb} & \longrightarrow & \min \\ \mathbf{y} & \geq & \mathbf{0} \end{array} \tag{2.1}$$

Here $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{b}$ is a column vector with m components and $\mathbf{c}$ is a row vector with n components. The matrix $\mathbf{A}$ and the vectors $\mathbf{b}$, $\mathbf{c}$ are constructed from the given graph $G = (V, E)$. The two programs are in a primal dual relation with each other.

In certain cases the vector $\mathbf{x}$ in the primal problem is restricted to have only 0-1 components. In other reformulations of the maximum clique problem the vector $\mathbf{y}$ in the dual problem is a 0-1 vector.

We will use the following observation to decide if a given LP reformulation is tighter than an other. Let us consider the following two constraints.

$$a_1 x_1 + \cdots + a_n x_n \leq b \tag{2.2}$$

$$a'_1 x'_1 + \cdots + a'_n x'_n \leq b' \tag{2.3}$$

LEMMA 2.1. If $b/a_i \leq b'/a'_i$ for each i, $1 \leq i \leq n$, then constraint (2.2) is tighter than constraint (2.3).

Proof. We form the $(n-1)$-dimensional hyperplanes

$$a_1 x_1 + \cdots + a_n x_n \leq b \tag{2.4}$$

$$a'_1 x'_1 + \cdots + a'_n x'_n \leq b' \tag{2.5}$$

associated with the constraints (2.2) and (2.3), respectively. The x_i-intercept of hyperplane (2.4) is equal to b/a_i. (If $a_i = 0$ then we set b/a_i to be equal to $+\infty$.) The x_i-intercept of hyperplane (2.5) is equal to b'/a'_i. (If b'/a'_i then we set b'/a'_i to be equal to $+\infty$.) If $b/a_i \le b'/a'_i$ for each i, $1 \le i \le n$, then for non-negative coordinates hyperplane (2.4) is closer to the origin of the coordinate system than hyperplane (2.5). □

2.1. The edge reformulation

Let $G = (V, E)$ be a finite simple graph with vertex set $V = \{v_1, \dots, v_n\}$. A node whose degree is equal to $n - 1$ is called a node with full degree. If v_i is a full degree node, then v_i is adjacent to each other vertex in G. In this situation we may delete the node v_i from G and work with the smaller graph. The cliques of the smaller graph can be augmented by v_i to get the cliques of the original graph G. We will suppose that the graph G does not contain any vertex with full degree.

Let

$$e_1 = (v_{f(1)}, v_{g(1)}), \dots, e_m = (v_{f(m)}, v_{g(m)})$$

be all the ordered pairs for which the corresponding unordered pairs

$$\{v_{f(1)}, v_{g(1)}\}, \dots, \{v_{f(m)}, v_{g(m)}\}$$

are not edges of the graph G. Here we assume that $f(i) < g(i)$ for each i, $1 \le i \le m$.

We construct an $m \times n$ matrix

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix}$$

by setting

$$a_{i,j} = \begin{cases} 1, & \text{if } j \in \{f(i), g(i)\} \\ 0, & \text{otherwise.} \end{cases}$$

Obviously, each row of $\mathbf{A}$ contains exactly two non-zero components. Each columns contains at least one non-zero element as we assumed that the graph G does not have any node with full degree.

We use the matrix $\mathbf{A}$ to form the following 0-1 linear program.

$$\mathbf{cx} \longrightarrow \max, \qquad \mathbf{Ax} \le \mathbf{b}. \tag{2.6}$$

Let Δ be a clique in G and let U be the set of nodes of Δ. Set

$$y_i = \begin{cases} 1, & \text{if } v_i \in U \\ 0, & \text{if } v_i \notin U. \end{cases}$$

LEMMA 2.2. The zero-one vector $\mathbf{y} = [y_1, \dots, y_n]^\top$ is a feasible solution of the linear program (2.6).

Proof. The (i)-th constraint of the linear program (2.6) is

$$a_{i,1}x_1 + \dots + a_{i,n}x_n \le 1. \tag{2.7}$$

Plug $\mathbf{y} = [y_1, \dots, y_n]^\top$ in place of $\mathbf{x}$ in $\mathbf{Ax} \le \mathbf{b}$ to get

$$a_{i,1}y_1 + \dots + a_{i,n}y_n \le 1. \tag{2.8}$$

from (2.7). The definition of the matrix $\mathbf{A}$ gives that $a_{i,f(i)} = a_{i,g(i)} = 1$ and the other coefficients in (2.7) are all equal to zero. Since the unordered pair $\{v_{f(i)}, v_{g(i)}\}$ is not an edge of G. The only case when the (i)-th constraint is violated when $y_{f(i)} = y_{g(i)} = 1$. In this situation the definition of Δ gives that the unordered pair $\{v_{f(i)}, v_{g(i)}\}$ is an edge of Δ and so is an edge of G. □

Let $\mathbf{y} = [y_1, \dots, y_n]^\top$ be a zero-one feasible solution of the linear program (2.6). Set $U = \{v_i : y_i = 1\}$. Let Δ be the subgraph of G induced by U.

LEMMA 2.3. The graph Δ is a clique in G.

TABLE 1. The adjacency matrices of the graphs G and G' in Example 2.4.

	1	2	3	4	5	6
1	×	•		•	•	
2	•	×	•			
3		•	×	•		•
4	•		•	×	•	
5	•			•	×	
6			•			×

	1	2	3	4	5	6
1	×		•			•
2		×		•	•	•
3	•		×		•	
4		•		×		•
5		•	•		×	•
6	•	•		•	•	×

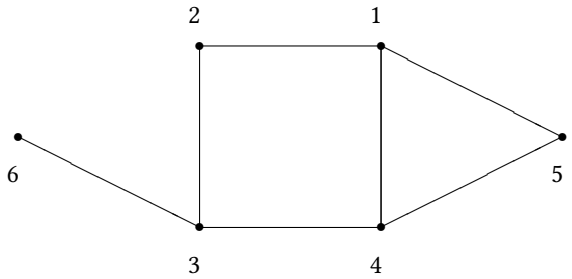

FIGURE 1. A geometrical representation of the graph G in Example 2.4.

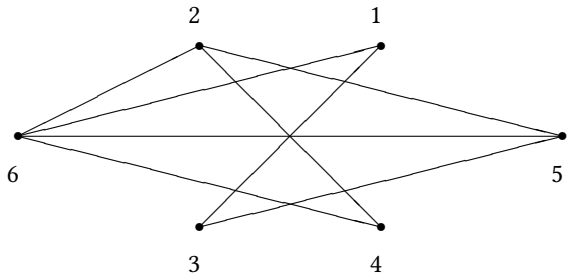

FIGURE 2. A graphical representation of the graph G' in Example 2.4.

Proof. In order to prove the lemma choose $v_k, v_l \in U$ such that $k < l$ and try to show that the unordered pair $\{v_k, v_l\}$ is an edge of G.

If $\{v_k, v_l\}$ is not an edge of G, then there is an index i such that $e_i = (v_k, v_l)$. In this situation $a_{i,k} = a_{i,l} = 1$ in the (i)-th constraint (2.7) of the linear program (2.6) and the other coefficients are all equal to zero. Let us plug $\mathbf{y} = [y_1, \dots, y_n]^\top$ in place of $\mathbf{x}$ in $\mathbf{Ax} \leq \mathbf{b}$ to get (2.8) from (2.7). It follows that either $y_k = 0$ or $y_l = 0$. Thus either $v_k \notin U$ or $v_l \notin U$. □

To illustrate the argument we worked out an example in details.

EXAMPLE 2.4. Let $G = (V, E)$ be the graph given by its adjacency matrix in Table 1. The graph has 6 nodes and 7 edges. A geometrical representation of G is depicted in Figure 1.

Let G' be the complement graph of G. The complement graph G' has 8 edges. The edges of G' can be described using a node-edge incidence matrix. The rows of this incidence matrix are labeled with the nodes of G' and the columns are labeled with the edges of G'. Table 2 contains the node-edge incidence matrix of G'. The idea of using bullets to represent non-zero entries in adjacency and incidence matrices is borrowed from [6].

We denote the edge-node incidence matrix of G' by $\mathbf{A}$ and using this matrix we formulate two linear programs (2.1). More detailed forms of the programs are presented in Table 3.

TABLE 2. The node-edge and edge-node incidence matrices of G' in Example 2.4.

	13	16	24	25	26	35	46	56
1	•	•						
2			•	•	•			
3	•					•		
4			•				•	
5				•		•		•
6		•			•		•	•

	1	2	3	4	5	6
1,3	•		•			
1,6	•					•
2,4		•		•		
2,5		•			•	
2,6		•				•
3,5			•		•	
4,6				•		•
5,6					•	•

TABLE 3. The primal and dual programs.

x_1	x_2	x_3	x_4	x_5	x_6	≥	0
1		1				≤	1
1					1	≤	1
	1		1			≤	1
	1			1		≤	1
	1				1	≤	1
		1		1		≤	1
			1		1	≤	1
				1	1	≤	1
1	1	1	1	1	1	⟶	max

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	≥	0
1	1							≥	1
		1	1	1				≥	1
1					1			≥	1
		1				1		≥	1
			1		1		1	≥	1
	1			1		1	1	≥	1
1	1	1	1	1	1	1	1	⟶	min

A 0-1 solution of the primal problem gives a packing of an 8 element set with the member of a set family which consists of 6 sets. A 0-1 solution of the primal problem corresponds to a subset U of V. The elements of U are the nodes of a clique in G. Note that no two distinct elements in U fall into the same color class of the nodes of G at a legal coloring of the nodes of G.

A 0-1 solution of the dual problem gives a covering of a 6 element set with the members of a set family consisting of 8 subsets. One can use a greedy set packing algorithm to locate a feasible solution x of the primal problem. In our case for instance we may choose $\mathbf{x}$ to be equal to $[1, 1, 0, 0, 0, 0]^\top$. Similarly, one can use a greedy set covering algorithm to find a feasible solution of the dual problem. Say, we may choose $\mathbf{y}$ to be $[1, 0, 1, 0, 0, 0, 0, 1]$. By the weak duality theorem,

$$\mathbf{cx} \leq (\mathbf{yA})\mathbf{x} = \mathbf{y}(\mathbf{Ax}) \leq \mathbf{yb}.$$

Here we may choose $\mathbf{y}$ to be a 0-1 solution. It means that a greedy covering can be used to establish upper estimates for the clique size of G. Relaxing the condition that $\mathbf{y}$ is a 0-1 vector shows that fractional coverings can be used to find upper estimates as well.

TABLE 4. The primal and dual programs.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	$\geq$	0
1	1							$\leq$	1
		1	1	1				$\leq$	1
1					1			$\leq$	1
		1				1		$\leq$	1
			1		1		1	$\leq$	1
	1			1		1	1	$\leq$	1
1	1	1	1	1	1	1	1	$\longrightarrow$	max

y_1	y_2	y_3	y_4	y_5	y_6	$\geq$	0
1		1				$\geq$	1
1					1	$\geq$	1
	1		1			$\geq$	1
	1			1		$\geq$	1
	1				1	$\geq$	1
		1		1		$\geq$	1
			1		1	$\geq$	1
				1	1	$\geq$	1
1	1	1	1	1	1	$\longrightarrow$	min

2.2. The edge covering set reformulation

Let $G = (V, E)$ be a finite simple graph. A subset H of V is called an edge covering set if each edge of G has at least one node in H. As an example consider the graph G' given in Figure 2. This graph has 6 nodes and $V = \{1, \ldots, 6\}$. The set $H = \{2, 3, 6\}$ is an edge covering set.

OBSERVATION 2.5. If H is an edge covering set of G, then $V \setminus H$ is an independent set of G.

Let U be a finite set and let $A_1, \ldots, A_n$ be subsets of U. We say that $A_1, \ldots, A_n$ form a covering system of U if their union is equal to U.

PROBLEM 2.6. Given a finite set U and a family of subsets F of U and given a positive integer k. Decide if F contains k members that form a covering system of U.

PROBLEM 2.7. Given a finite set U and a family of subsets F of U. Find the minimum of k for which F contains a covering system of U.

Problem 2.6 is referred to as the k-covering problem and Problem 2.7 is called the minimum set covering problem. To decide if the family F contains a covering system of U is an easy task. One should compute the union of the members of F and check if this is equal to U. On the other hand the k-covering problem is hard. It is plain that an efficient algorithm that solves the k-covering problem can be used to solve the minimum set covering problem.

By Observation 2.5, if H is an edge covering set of G, then the set $V \setminus H$ induces a clique in G'. The clique search problem can be reduced to a set covering problem. Let $G = (V, E)$ be a given graph and suppose we are looking for a maximum clique in G. First we form the complement G' of the graph G. Then we try to locate a minimum edge covering set H in V. The set $V \setminus H$ is a maximum independent set in G' and so $V \setminus H$ induces a maximum clique in G.

The incidence matrix of the set covering problem in Example 2.4 is given in Table 2. The ground set which should be covered is the set formed by all the edges of G'. The rows of the incidence matrix are labeled by the nodes of G' and the columns of the incidence matrix are labeled by the edges of G'. When v is a node of G' and e is an edge of G', then we put a bullet into the cell at the intersection of the row v and the column e.

Replacing x_i by $1 - x_i'$ in the primal program in Table 4 we get back the dual program presented in Table 3. Similarly, replacing y_j by $1 - y_j'$ in the dual program in Table 4 we get back the primal program depicted in Table 3.

A 0-1 solution of the primal program in Table 4 can be interpreted as a matching in the graph G'. The size of a maximum matching in G' is called the matching number of G' and it is denoted by $\mu(G')$. Note that

$$\mu(G') = \mathbf{cx} \leq \mathbf{yb} = n - \omega(G),$$

where n is the number of nodes of G. This is equivalent to $\omega(G) \leq n - \mu(G')$. Expressing the result in plain English we can say that a lower bound of the matching number of the complement graph G' can be used to get an upper estimate for the clique number of the original graph G. The inequality $\omega(G) \leq n - \mu(G')$ is not a very good upper bound for the clique size of $\omega(G)$. Namely, $\mu(G')$ cannot be larger than $n/2$ and so the upper bound for $\omega(G)$ cannot be smaller than $n/2$.

2.3. The symmetric reformulation

In this section we describe the 0-1 linear programming reformulation of the maximum clique problem given by F. D. Croce and R. Tadei [3]. We give new justification of this approach. The reason is the following. Firstly, in this way all results in paper will have similar proofs. Secondly, at the end of section 3.2 we will point out how this reformulation can be tightened. The reader can go through the proofs in this section and can verify that the modified versions are correct.

Suppose $v_1, \ldots, v_n$ are all the nodes of the graph $G = (V, E)$ and let $G' = (V', E')$ be the complement graph of G. Let

$$a_{i,1}x_1 + \cdots + a_{i,n}x_n \leq b_i \tag{2.9}$$

be the (i)-th constraint. We set

$$a_{i,j} = \begin{cases} 1, & \text{if } \{v_i, v_j\} \in E' \\ 0, & \text{if } \{v_i, v_j\} \notin E' \end{cases}$$

Then we set $a_{i,i}$ and b_i both to be $n - \deg(v_i) - 1$, where $\deg(v_i)$ is the degree of the node v_i in the graph G. The matrix

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \ldots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \ldots & a_{n,n} \end{bmatrix}$$

is symmetric. So in the lack of standard terminology let us call this reformulation of the maximum clique problem the symmetric reformulation.

Let Δ be a clique in G and let U be the set of nodes of Δ. Set

$$y_i = \begin{cases} 1, & \text{if } v_i \in U, \\ 0, & \text{if } v_i \notin U. \end{cases}$$

PROPOSITION 2.8. The 0-1 vector $\mathbf{y} = [y_1, \ldots, y_n]^\top$ satisfies the constraint (2.9) for each i, $1 \leq i \leq n$.

Proof. Plug $\mathbf{y} = [y_1, \ldots, y_n]^\top$ into (2.9) for each i, $1 \leq i \leq n$. Let us consider the (i)-th constraint. In order to prove the claim assume first that $y_i = 1$. This means that $v_i \in U$. Note that in the (i)-th constraint $a_{i,i} = b_i = n - \deg(v_i) - 1$. Thus the (i)-th constraint reduces to

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j}y_j \leq 0. \tag{2.10}$$

The (i)-th constraint is not satisfied only if there is an index j such that $y_j = 1$, $a_{i,j} = 1$, $1 \leq j \leq n, j \neq i$. As $y_i = 1$, $y_j = 1$, by the definition of $\mathbf{y} = [y_1, \ldots, y_n]$ it follows that $v_i, v_j \in U$. As Δ is a clique in G and $v_i, v_j \in U$, we get that $\{v_i, v_j\} \in E$ and so $a_{i,j} = 0$. Therefore, the (i)-th constraint is satisfied when $y_i = 1$.

TABLE 5. The symmetric and the triangular shape linear programs.

x_1	x_2	x_3	x_4	x_5	x_6	$\geq$	0
2		1			1	$\leq$	2
	3		1	1	1	$\leq$	3
1		2		1		$\leq$	2
	1		2		1	$\leq$	2
	1	1		3	1	$\leq$	3
1	1		1	1	4	$\leq$	4
1	1	1	1	1	1	$\longrightarrow$	max

x_1	x_2	x_3	x_4	x_5	x_6	$\geq$	0
2		1			1	$\leq$	2
	3		1	1	1	$\leq$	3
		1		1		$\leq$	1
			1		1	$\leq$	1
				1	1	$\leq$	1
					1	$\leq$	1
1	1	1	1	1	1	$\longrightarrow$	max

Let us deal with the $y_i = 0$ case next. The condition $y_i = 0$ means that $v_i \notin U$. Using $y_i = 0$ the (i)-th constraint can be written in the form

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} y_j \leq b_i.$$

On the left hand side of the inequality $a_{i,j} = 1$ can hold for at most $n - \deg(v_i) - 1$ choices of the index j. Thus the (i)-th constraint is satisfied when $y_j = 0$. □

Let $\mathbf{y} = [y_1, \ldots, y_n]^\top$ be a 0-1 solution of the constraint (2.9) for each $i, 1 \leq i \leq n$. Set $U = \{v_i : y_i = 1\}$. Let Δ be the subgraph of G induced by U.

PROPOSITION 2.9. The graph Δ is a clique in G.

Proof. In order to prove the claim choose $v_i, v_j \in U$ such that $v_i \neq v_j$ and try to show that $\{v_i, v_j\}$ is an edge of G. As $y_i = 1$, the (i)-th constraint reduces to (2.10). If $\{v_i, v_j\} \notin E$, then $a_{i,j} = 1$ and we get the contradiction

$$1 \leq \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{i,j} y_j \leq 0.$$

Thus $\{v_i, v_j\}$ is an edge of G. □

Let us see a small size numerical example. We start with the graph G given in Example 2.4. The graph G has 6 nodes $v_1 = 1, \ldots, v_6 = 6$ whose degrees are 3, 2, 3, 3, 2, 1, respectively. The complement graph G' of G has the same nodes but the degrees of the nodes in G' are 2, 3, 2, 2, 3, 4, respectively. We introduce six 0-1 variables $x_1, \ldots, x_6$ and form the linear program given in Table 4.

Here is how we set up this linear program. The coefficients of the objective functions are all set to be 1. The right hand sides of the inequalities are filled with the degrees of the nodes of G'. Similarly, the main diagonal is filled with the degrees of the nodes of G'. In G' node v_1 is connected with v_3 and v_6. Therefore we set the coefficients of x_3 and x_6 to be 1 in the (1)-st constraint. The node v_2 is connected with v_4, v_5, v_6 in G' and so we set the coefficients of x_4, x_5, x_6 to be 1 in the (2)-nd constraint. Continuing in this way we ended up with 6 constraints corresponding to the 6 nodes of G'.

2.4. The triangular shape reformulation

We present a further 0-1 linear programming reformulation of the maximum clique problem.

Let $G = (V, E)$ be a finite simple graph such that $V = \{v_1, \dots, v_n\}$. We form a sequence of graphs $G_1, \dots, G_n$. Here $G_1 = G$. We construct G_2 from G_1 by deleting the node v_1. We get G_3 from G_2 by deleting the node v_2. In general, we get G_{i+1} from G_i by deleting the node v_i for each i, $1 \le i \le n-1$. Let H_i be the subgraph of $G_i = (V_i, E_i)$ induced by the set of nodes of $V_i \setminus \{v_i\}$ that are not connected to v_i. Suppose b_i is an upper bound of $\omega(H_i)$, that is, $\omega(H_i) \le b_i$. The (i)-th constraint of the linear program is

$$a_{i,i}x_i + a_{i,i+1}x_{i+1} + \cdots + a_{i,n}x_n \le b_i, \tag{2.11}$$

where $a_{i,i} = b_i$ and

$$a_{i,j} = \begin{cases} 0, & \text{if } \{v_i, v_j\} \in E \\ 1, & \text{if } \{v_i, v_j\} \notin E \end{cases}$$

for each i, j, $1 \le i < j \le n$.

Let Δ be a clique in G such that U is the set of nodes of Δ. Set

$$y_i = \begin{cases} 1, & \text{if } v_i \in U, \\ 0, & \text{if } v_i \notin U. \end{cases}$$

PROPOSITION 2.10. The 0-1 vector $\mathbf{y} = [y_1, \dots, y_n]^\top$ satisfies (2.11) for each i, $1 \le i \le n-1$.

Proof. Plug $\mathbf{y} = [y_1, \dots, y_n]^\top$ to (2.11) and consider the (i)-th constraint. Suppose first that $y_i = 1$. Now the (i)-th constraint reduces to

$$\sum_{j=i+1}^{n} a_{i,j}y_j \le 0. \tag{2.12}$$

The (i)-th constraint is not satisfied only if there is an index j for which $y_j = 1$, $a_{i,j} = 1$, $i+1 \le j \le n$. As $y_i = 1$, $y_j = 1$, by the definition of $\mathbf{y} = [y_1, \dots, y_n]^\top$, it follows that $v_i, v_j \in U$. Since Δ is a clique in G and $v_i, v_j \in U$, it follows that $\{v_i, v_j\} \in E_i$. Consequently, $a_{i,j} = 0$. Therefore, the (i)-th constraint is satisfied when $y_i = 1$.

Next let us deal with the case when $y_i = 0$. Now the (i)-th constraint reduces to

$$\sum_{j=i+1}^{n} a_{i,j}y_j \le b_i.$$

On the left hand side of the inequality $a_{i,j} = 1$ can hold for at most b_i choices of the index j. Thus the i-th constraint is satisfied when $y_i = 0$. □

Let $\mathbf{y} = [y_1, \dots, y_n]^\top$ be a 0-1 solution of (2.11) for each i, $1 \le i \le n$. Set $U = \{v_i : y_i = 1\}$ and let Δ be the graph induced by U in G.

PROPOSITION 2.11. The graph Δ is a clique in G.

Proof. Choose $v_i, v_j \in U$ such that $v_i \ne v_j$ and try to show that $\{v_i, v_j\} \in E$. We may assume that $j > i$ since this is only a matter of exchanging v_i and v_j. As $v_i \in U$, it follows that $y_i = 1$. Now the (i)-th constraint reduces to (2.12). If $\{v_i, v_j\} \notin E$, then $\{v_i, v_j\} \notin E_i$ and so $a_{i,j} = 1$. Thus we get the contradiction

$$1 \le \sum_{j=i+1}^{n} a_{i,j}y_j \le 0.$$

Therefore $\{v_i, v_j\}$ is an edge of G. □

We illustrate the above arguments in the particular case when the graph G is given in Example 2.4 the corresponding linear program can be seen in Table 5.

Here is how we constructed the linear program. We started with the adjacency matrix of G in Table 1. The degree of v_1 in G' (in the complement of G) is equal to 2. This number 2 stands in the main diagonal and the right hand side in the (1)-st constraint. The coefficient of x_3, x_6 are equal to 1 since v_3, v_6 are the neighbors of v_1 in G'.

TABLE 6. The adjacency matrix and a rearranged adjacency matrix of the graph G in Example 3.1.

	1	2	3	4	5	6	7	8	9	10	11	12
1	×	•							•		•	•
2	•	×			•		•	•		•	•	•
3			×	•		•			•			•
4			•	×				•				
5		•			×	•	•	•		•		
6			•		•	×	•	•			•	
7		•			•	•	×	•		•		
8		•		•	•	•	•	×		•		
9	•		•						×		•	
10		•			•		•	•		×		•
11	•	•				•			•		×	•
12	•	•	•							•	•	×

	4	3	9	1	1	2	6	2	5	7	8	10
4	×	•									•	
3	•	×	•			•	•					
9		•	×	•	•							
1			•	×	•	•		•				
11			•	•	×	•	•	•				
12		•		•	•	×		•				•
6		•			•		×		•	•	•	
2				•	•	•		×	•	•	•	•
5							•	•	×	•	•	•
7							•	•	•	×	•	•
8	•						•	•	•	•	×	•
10						•		•	•	•	•	×

We construct a new graph G_2 from G by deleting the node v_1. For a unified notation we denote G by G_1. Of course, the adjacency matrix of G_2 can be constructed from the adjacency matrix of G_1 by deleting the row and column labeled by v_1.

The neighbors of v_2 in G_2' are v_4, v_5, v_6. Thus we put a 3 in the main diagonal and the right hand side in the (2)-nd constraint. The coefficients of x_4, x_5, x_6 will be 1. We construct a new graph G_3 from G_2 by deleting the node v_2.

The only neighbor of v_3 in G_3' is v_5. Thus we put a 1 in the main diagonal and the right hand side in the (3)-rd constraint. The coefficient of x_5 will be 1. We construct a new graph G_4 from G_3 by deleting the node v_3.

The only neighbor of v_4 in G_4' is v_6. Thus we put a 1 in the main diagonal and the right hand side in the (4)-th constraint. The coefficient of x_6 will be 1. We construct a new graph G_5 from G_4 by deleting the node v_4.

The only neighbor of v_5 in G_5' is v_6. Thus we put a 1 in the main diagonal and the right hand side in the (5)-th constraint. The coefficient of x_6 will be 1. We construct a new graph G_6 from G_5 by deleting the node v_5. The graph G_6 consists of a single node and so the procedure terminates.

3. PRECONDITIONING

A preliminary inspection and modification of an optimization problem many times pays off handsomely resulting a more manageable version of the original problem. In this section we describe such preconditioning in connection with the edge reformulation and in connection with the triangular shape reformulation.

The preconditioning procedure cannot be well illustrated with a small graph like the one in Example 2.4 so we consider a larger but still toy size example.

EXAMPLE 3.1. Let us consider the graph $G = (V, E)$ given by its adjacency matrix in Table 6. The graph G has 12 nodes and 28 edges. A possible geometric representation of G can be seen in Figure 3. The nodes of G are denoted by $v_1 = 1, \ldots, v_{12} = 12$.

3.1. Condensing the constraints

We will add and remove constraints in the edge formulation of the maximum clique problem. As a result the number of constraints will decrease. In the same time the upper bound provided by the real relaxation of the problem gets tighter.

A greedy sequential coloring of the nodes of G gives the coloring in Table 7.

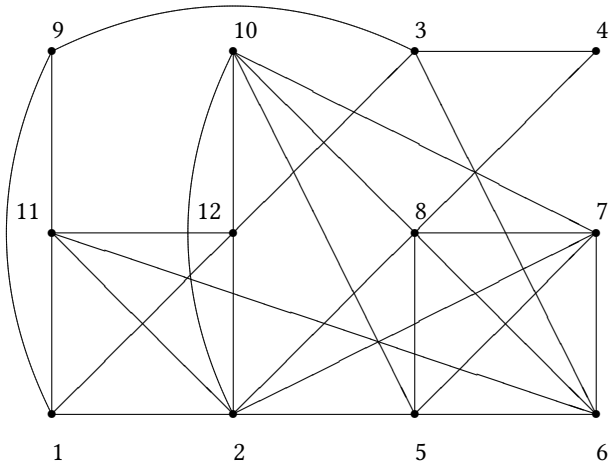

FIGURE 3. A graphical representation of the graph G in Example 3.1.

TABLE 7. Color classes provided by the greedy sequential coloring in Example 3.1.

color class	C_1	C_2	C_3	C_4	C_5
	1	2	7	8	10
	3	4	11	12	
	5	6			
		9			

color class	C_1	C_2	C_3	C_4	C_5
	1	2	5	6	8
	4	3	9	10	11
	7		12		

The color class $C_1 = \{1, 3, 4\}$ contains the information that the pairs $\{1, 3\}$, $\{1, 5\}$, $\{3, 5\}$ are not edges of G.

The color class $C_2 = \{2, 4, 6, 9\}$ contains the information that the pairs $\{2, 4\}$, $\{2, 6\}$, $\{2, 9\}$, $\{4, 6\}$, $\{4, 9\}$, $\{6, 9\}$ are not edges of G.

The color class $C_3 = \{7, 11\}$ carries the information that the pair $\{7, 11\}$ is not an edge of G.

The color class $C_4 = \{8, 12\}$ codes the information that the pair $\{8, 12\}$ is not an edge of G.

The color class $C_5 = \{10\}$ does not give any information about the edges of G.

We add the listed 11 pairs to the graph G as edges. We denote the resulting graph by G_2. For the sake of a uniform notation we denote G by G_1. Continuing in this way we end up with 8 graphs $G_1, \dots, G_8$. The color classes of the coloring of these graphs contains all the information about the 2 element subsets of V that are not edges of G. In fact we need only the color classes that contain at least two nodes. The color classes are listed in Table 8.

Using these 24 color classes we can set up a 0-1 linear program which is equivalent of the maximum clique problem. Table 9 contains the linear program. The edge reformulation of the problem contains 34 inequalities. The point is that the new reformulation has less inequalities. In addition the new reformulation is tighter.

For instance the first constraint $x_1 + x_3 + x_5 \leq 1$ in the condensed reformulation in Table 9 replaces the constraints $x_1 + x_3 \leq 1$, $x_1 + x_5 \leq 1$, $x_3 + x_5 \leq 1$ in the edge reformulation. The x_1, x_3, x_5 intercepts of the hyperplane $x_1 + x_3 + x_5 = 1$ are 1 and the other intercepts are $+\infty$. The x_1, x_3 intercepts of the

TABLE 8. Color classes provided by the greedy sequential colorings in Example 3.1.

G_1	$\{1,3,5\}$	$\{2,4,6,9\}$	$\{7,11\}$	$\{8,12\}$	
G_2	$\{1,4,7\}$	$\{2,3\}$	$\{5,9,12\}$	$\{6,10\}$	$\{8,11\}$
G_3	$\{1,6\}$	$\{3,7\}$	$\{4,5,11\}$	$\{8,9\}$	
G_4	$\{1,8\}$	$\{3,10,11\}$	$\{4,12\}$	$\{7,9\}$	
G_5	$\{1,10\}$	$\{3,8\}$	$\{6,12\}$		
G_6	$\{4,10\}$	$\{7,12\}$			
G_7	$\{1,9\}$				
G_8	$\{9,10\}$				

TABLE 9. The condensed edge reformulation in Example 3.1.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	$\geq$	0
1		1		1								$\leq$	1
	1		1		1			1				$\leq$	1
						1				1		$\leq$	1
							1				1	$\leq$	1
1			1			1						$\leq$	1
	1	1										$\leq$	1
				1				1			1	$\leq$	1
					1				1			$\leq$	1
							1			1		$\leq$	1
1					1							$\leq$	1
		1				1						$\leq$	1
			1	1						1		$\leq$	1
							1	1				$\leq$	1
1							1					$\leq$	1
		1							1	1		$\leq$	1
			1								1	$\leq$	1
						1		1				$\leq$	1
1									1			$\leq$	1
		1					1					$\leq$	1
					1						1	$\leq$	1
			1						1			$\leq$	1
						1					1	$\leq$	1
1								1				$\leq$	1
								1	1			$\leq$	1
1	1	1	1	1	1	1	1	1	1	1	1	$\rightarrow$	max

hyperplane $x_1 + x_3 = 1$ are 1 and the other intercepts are $+\infty$. The x_1, x_5 intercepts of the hyperplane $x_1 + x_5 = 1$ are 1 and the other intercepts are $+\infty$. The x_3, x_5 intercepts of the hyperplane $x_3 + x_5 = 1$ are 1 and the other intercepts are $+\infty$. By Lemma 2.1, the constraint $x_1 + x_3 + x_5 \leq 1$ is tighter than each of the constraints $x_1 + x_3 \leq 1$, $x_1 + x_5 \leq 1$, $x_3 + x_5 \leq 1$.

The set of the feasible solutions of the real relaxation of the edge reformulation contains the set of feasible solutions of the real relaxation of the new reformulation.

It seems that we could proceed by finding an independent set C in G. The independent set C codes a number of 2 element subsets of V that are not edges of G. We construct a new graph G_2 from G by adding these pairs to G as edges. Then we locate an independent set in G_2. This approach offers certain advantages. We can use any greedy technique to locate an independent set. For instance we can use the Carraghan-Pardalos algorithm to locate an independent set. One should feed the

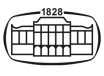

Table 10. The triangular shape reformulation in Example 3.1.

x_4	x_3	x_9	x_1	x_{11}	x_{12}	x_6	x_2	x_5	x_7	x_8	x_{10}	$\geq$	0
9		1	1	1	1	1	1	1	1		1	$\leq$	9
	7		1	1			1	1	1	1	1	$\leq$	7
		7			1	1	1	1	1	1	1	$\leq$	7
			5			1		1	1	1	1	$\leq$	5
				4				1	1	1	1	$\leq$	4
					4	1		1	1	1		$\leq$	4
						2	1				1	$\leq$	2
							1					$\leq$	1
								1				$\leq$	1
									1			$\leq$	1
										1		$\leq$	1
											1	$\leq$	1
1	1	1	1	1	1	1	1	1	1	1	1	$\longrightarrow$	max

complement of G into the algorithm. In addition, we do not need to prove the optimality of the independent set. Any suboptimal independent set will do.

3.2. Reordering the nodes

Setting up the triangular shape reformulation of the maximum clique problem depends on the order of the nodes of G. Permuting the nodes leads to a different reformulation. One possibility of fixing the order of the nodes is the following. Choose v_1 to be a node of G whose degree is minimal in G. By deleting v_1 we get G_2. Choose v_2 to be a node of G_2 whose degree is minimal in G_2. Continuing in this way finally we get a sequence of nodes $v_1, \dots, v_n$ and a sequence of graphs $G_1, \dots, G_n$. We illustrated the procedure in connection with the graph G in Example 3.1. We get the permutation

$$v_4, v_3, v_9, v_1, v_{11}, v_{12}, v_6, v_2, v_5, v_7, v_8, v_{10}$$

of the nodes $v_1, \dots, v_{12}$. The linear program can be seen in Table 9. In order to make the reader's life easier we included an adjacency matrix of G whose rows and columns are rearranged corresponding to the above permutation in Table 6.

Let us note that the nodes $v_2, v_5, v_7, v_8, v_{10}$ are nodes of a 5-clique in G. In general this procedure always ends up with a clique in G. In other words we always get a lower estimate for the clique size of G as a bonus.

The upper estimate b_i of the clique number of H_i can be chosen in many ways. The most trivial is to choose b_i to be the number of nodes of H_i. As a less trivial approach one may color the nodes of H_i to construct an upper bound for $\omega(H_i)$. The Carraghan-Pardalos algorithm can also be used to produce upper bounds for the clique number of H_i.

In order to illustrate the remark above we used greedy sequential colorings of the nodes of the graph H_i to estimate b_i instead of the size of H_i. In this way we could replace the numbers 9, 7, 7, 5, 4, 4, 2 in the main diagonal (and on the right hand side as well) by the numbers 5, 5, 6, 4, 4, 4, 2, respectively. As a result the optimal solution of the real relaxation of the program went down from 6.32 to 5.33.

The 1-st constraint of the linear program is bounded by a 9-dimensional hyperplane. The intercepts of the coordinate axes

$$x_4, x_3, x_9, x_1, x_{11}, x_{12}, x_6, x_2, x_5, x_7, x_8, x_{10}$$

and the hyperplane are

$$1, +\infty, 9, 9, 9, 9, 9, 9, 9, 9, +\infty, 9$$

TABLE 11. The labels of the columns in the tables summarizing the numerical experiments.

label	reformulation	local coloring	global coloring
e	edge		
eC	edge		yes
s	symmetric		
sc	symmetric	yes	
sC	symmetric		yes
scC	symmetric	yes	yes
t	tringular		
tc	tringular	yes	
tC	tringular		yes
tcC	tringular	yes	yes

when we use the number of the nodes of the graph H_i to estimate b_i. On the other hand, the intercepts are

$$1, +\infty, 5, 5, 5, 5, 5, 5, 5, 5, +\infty, 5$$

when we use the number of the color classes of the graph H_i to estimate b_i. By Lemma 2.1, the second hyperplane is closer to the origin of the coordinate system than the first hyperplane for non-negative coordinates. This holds in general for each constraint. Consequently, the polyhedron of the feasible solutions in the second case is always inside the polyhedron of the feasible solutions in the first case.

4. NUMERICAL EXPERIMENTS

Suppose we are working with the edge reformulation in order to determine the clique size of a given graph G. The real relaxation of the 0-1 linear program gives an upper estimate of the clique size $\omega(G)$ of G. If the graph G has n nodes and it does not have any node of full degree, then setting each variable to be 1/2 we get a feasible solution. Therefore the upper estimate of $\omega(G)$ cannot be better than $n/2$. Augmenting the linear program with constraints corresponding to the color classes of a legal coloring of the nodes of G improves this upper estimate provided that the color classes have many elements. We will refer to this maneuver as adding a global coloring to the linear program. Of course this can be done in connection with the symmetric and triangular shape reformulations.

In the (i)-th constraint in the standard version of the symmetric reformulation the coefficients $a_{i,j}$, b_i were set to be $n - 1 - \deg(v_i)$. The essential point of this choice is that the quantity $n - 1 - \deg(v_i)$ is an upper bound of $\omega(H_i)$. Here H_i is the subgraph of G induced by the non-neighbors of the node v_i. Naturally a legal coloring of the nodes of H_i provides an upper estimate of $\omega(H_i)$. We will refer to this modification of the standard symmetric reformulation as applying local colorings. Needles to say that one can use local colorings to improve the standard triangle shape reformulation as well.

In connection with a given graph we may formulate 10 linear programs. These possibilities are listed in Table 11. The question is inevitable. Which of these reformulations is the most advantageous. We picked 22 DIMACS bench mark graphs and considered 10 linear programs associated with each of these test graphs. In Tables 12 and 13 we listed the upper estimates given by these 220 linear programs.

The numerical evidence seems to support the conclusion that the symmetric reformulation with local and global colorings gives the best upper bound among the 10 competing reformulations.

TABLE 12. Numerical results in connection with the edge and symmetric reformulations.

graph	eC	e	scC	sc	sC	s
brock200_1	57.33	100.0	54.97	56.47	58.53	100.0
brock200_2	35.33	100.0	33.84	35.11	35.94	100.0
brock400_2	99.00	200.0	96.10	99.91	99.94	200.0
brock400_4	99.00	200.0	95.55	100.25	99.93	200.0
c-fat500-5	64.00	250.0	64.00	67.93	64.00	250.0
c-fat500-10	126.00	250.0	126.00	143.46	126.00	250.0
hamming8-2	128.00	128.0	128.00	128.00	128.00	128.0
hamming8-4	32.00	128.0	32.00	37.93	32.00	128.0
hamming10-4	128.00	512.0	128.00	145.57	128.00	512.0
johnson8-2-4	6.00	14.0	4.00	4.00	6.00	14.0
johnson16-2-4	14.00	60.0	8.00	8.00	14.00	60.0
keller4	37.00	85.5	32.21	34.19	37.00	85.5
keller5	175.00	388.0	134.02	138.50	175.00	388.0
keller6	781.00	1680.5	-	-	-	-
p_hat300-2	54.33	150.0	53.53	56.94	55.88	150.0
p_hat700-1	52.83	350.0	51.29	52.47	52.99	350.0
p_hat700-2	111.00	350.0	111.14	119.70	113.87	350.0
p_hat1500-1	95.00	750.0	-	-	-	-
san200_0.9_3	71.00	100.0	71.15	80.56	72.42	100.0
san200_0.9_2	83.50	100.0	81.09	89.65	84.83	100.1
sanr200_0.9	78.50	100.0	76.37	78.64	80.67	100.0
sanr200_0.7	51.67	100.0	48.99	50.93	51.98	100.0

TABLE 13. Numerical results in connection with the triangular shape reformulation.

graph	tcC	tc	tC	t
brock200_1	58.33	61.56	59.00	100.54
brock200_2	36.00	38.00	36.00	100.20
brock400_3	99.84	107.48	100.00	200.58
brock400_4	99.82	109.07	99.91	200.63
c-fat500-5	64.00	68.21	64.00	250.02
c-fat500-10	126.00	143.81	126.00	250.09
hamming8-2	128.00	128.00	128.00	128.00
hamming8-4	32.00	38.26	32.00	128.00
hamming10-4	128.00	145.78	128.00	512.00
johnson8-2-4	5.76	6.46	6.00	14.00
johnson16-2-4	12.79	14.59	14.00	60.00
keller4	32.11	38.09	37.00	85.63
keller5	137.43	164.82	175.00	388.69
keller6	-	-	-	-
p_hat300-2	55.97	64.91	55.99	155.96
p_hat700-1	53.00	57.73	53.00	351.11
p_hat700-2	113.98	136.24	113.99	362.63
p_hat1500-1	-	-	-	-
san200_0.9_3	72.62	85.72	72.80	102.68
san200_0.9_2	84.49	95.95	86.00	104.31
sanr200_0.9	79.82	85.37	81.85	103.46
sanr200_0.7	52.00	56.40	52.00	100.73

REFERENCES

[1] Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. *The Maximum Clique Problem, Handbook of Combinatorial Optimization*, vol. 4. Kluwer Academic Publisher, 1999.

[2] Carraghan, R., and Pardalos, P. M. An exact algorithm for the maximum-clique problem. *Operation Research Letters 9* (1990), 375–382.

[3] Croce, F. D., and Tadei, R. A multi-kp modelling for the maximum-clique problem. *European Journal of Operational Research 73* (1994), 555–561.

[4] Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, New York, 2003.

[5] Hasselberg, J., Pardalos, P. M., and Vairaktarakis, G. Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization 3* (1993), 463–482. http://www.springerlink.com/content/p2m65n57u657605n.

[6] Kárteszi, F. *Introduction to Finite Geometries*. North-Holland Pub. Co., Amsterdam, 1976.

[7] Konc, J., and Janežič, D. An improved branch and bound algorithm for the maximum clique problem. *MATCH Communications in Mathematical and Computer Chemistry 58* (2007), 569–590.

[8] Kumlander, D. *Some Practical Algorithms to Solve the Maximal Clique Problem*. PhD thesis, Tallin University of Technology, 2005.

[9] Östergård, P. R. J. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics 120* (2002), 197–207.

[10] Papadimitriou, C. H. Addison-Wesley Publishing Company, Inc., Reading, MA, 1994.

[11] Tomita, E., and Seki, T. An efficient branch-and-bound algorithm for finding a maximum clique. In *Lecture Notes in Computer Science*, vol. 2631. 2003, pp. 278–289.