Central European Journal of Operations Research (2022) 30:1353–1367
https://doi.org/10.1007/s10100-021-00776-z

Numerical experiments with LP formulations of the maximum clique problem

Dóra Kardos[1] · Patrik Patassy[1] · Sándor Szabó[2] · Bogdán Zaválnij[3]

Accepted: 19 August 2021 / Published online: 5 September 2021

Abstract
The maximum clique problems calls for determining the size of the largest clique in a given graph. This graph problem affords a number of zero-one linear programming formulations. In this case study we deal with some of these formulations. We consider ways for tightening the formulations. We carry out numerical experiments to see the improvements the tightened formulations provide.

Keywords Combinatorial optimization · Maximum clique problem · Zero-one linear programming · Greedy coloring · Practical solutions of NP complete problems · LP relaxation bounds

1 Introduction

All graphs appearing in this paper are finite simple graphs. In other words graphs have finitely many nodes and finitely many edges. They do not have loops and double edges. Let $G = (V, E)$ be a finite simple graph, where V is the set of nodes and E is the set of edges of G. 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

✉ Bogdán Zaválnij
bogdan@renyi.hu

Dóra Kardos
kaduabt@hotmail.com

Patrik Patassy
p.patrik08@outlook.com

Sándor Szabó
sszabo7@hotmail.com

1 University of Szeged, Szeged, Hungary

2 Institute of Mathematics and Informatics, University of Pecs, Pecs, Hungary

3 Alfred Renyi Institute of Mathematics, Budapest, Hungary

Δ is a k-clique in G. A k-clique Δ in G is called a maximum clique if G does not contain any $(k+1)$-clique. A k-clique Δ in G is called a maximal clique if Δ is not a subgraph of any $(k+1)$-clique in G. Each maximum clique in G has the same size. This well defined number is called the clique number of G and it is denoted by $\omega(G)$. The following two problems are known as the k-clique problem and the maximum clique problem, respectively.

Problem 1 Given a finite simple graph G and given a positive integer k. Decide if G has a clique of size k.

Problem 2 Given a finite simple graph G. Determine the size of the maximum cliques in G.

From the complexity theory of computation we know that these problems are computationally hard. Namely, Problem 1, as a decision problem, belongs to the NP complete complexity class. (For details see Garey and Johnson 2003 or Papadimitriou 1994.) 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 Bomze et al. (1999) and many bench mark problems are given in Hasselberg et al. (1993).

The work horses in the majority of the real life clique search computations are the Carraghan and Pardalos (1990) and the Östergård (2002) algorithms. Equipped with pruning methods coming from elementary combinatorial considerations such as coloring and matching, well tuned implementations of these algorithms are capable of handling highly non-trivial instances. (See for example Konc and Janežič 2007; Kumlander 2005; Tomita and Seki 2003.)

The maximum clique problem can be expressed as a linear program. In fact, there are various linear programming (LP) reformulations of the maximum clique problem. When a clique search instance falls out of the range of the combinatorial type algorithms we may try to deploy the LP machinery. We may consider a sufficiently small subgraph H of the given graph G. Applying a combinatorial algorithm to H we may gather information that can be added to the linear program associated with G as a cutting plane. In other words we may probe judiciously chosen subgraphs H of G and test them by combinatorial algorithms. The LP machinery is then used to aggregate these partial results to get an upper estimate of the clique number of the graph G. On the other hand, if the graph is not big, one can use an integer linear programming (ILP) solver for finding exact solutions as well. It is an empirical observation that typically if the upper estimates are better, then the ILP solver can solve the problem faster. In the paper we report on the results of these type of numerical experiments. Our paper is essentially a case study, to compare the merits of various approaches by means of numerical experiments.

The structure of the paper as follows. In Sect. 2 we describe the canonical formulations of the maximum clique problem. In Sect. 3 we analyze these formulations and their relations to each other. In Sect. 4 we propose new methods for tightening these formulations. In Sect. 5 we introduce new cuts based on s-clique free node sets. In Sect. 6 we propose a method for choosing which variable should be set binary and which real for mixed integer programming (MIP) approach solution. The last section is summarizing the result of extended numerical measurements of affect of the described and proposed methods.

2 Canonical 0–1 LP reformulations

In this section we describe some 0–1 linear programming equivalents for the maximum clique problem we used. We have to mention that there are further 0–1 LP reformulations of the maximum clique problem. Also we have to point out that these results are not new and we have compiled them merely for the convenience of the reader.

Let $G = (V, E)$ be a finite simple graph, where $V = \{v_1, \ldots, v_n\}$. Let Δ be a clique in G and let U be the set of nodes of Δ. We introduce decision variables $x_1, \ldots, x_n$, $x_i \in \{0, 1\}$. Here

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

for each i, $1 \leq i \leq n$. The optimum value of the linear program

$$\begin{aligned} &x_1 + \cdots + x_n \to \max \\ &x_i + x_j \leq 1, \quad \text{for } \{v_i, v_j\} \notin E \end{aligned}$$

gives the clique number of the graph G. This is the so-called edge reformulation and it is the most commonly encountered reformulation. The linear program

$$\begin{aligned} &x_1 + \cdots + x_n \to \max \\ &\textstyle\sum_{i \in I} x_i \leq 1, \text{ for each independent set } I \text{ of } G \end{aligned}$$

is known as the independent set reformulation of the maximum clique problem. In practice instead of listing all independent sets only maximal independent sets are listed, which is an equivalent formulation. The question about minimizing the listed independent sets while keeping the correctness was discussed in Beke et al. (2021).

The above reformulations are part of the folklore. For the next reformulation we need to introduce some notations. For a node v_j of G we define the set $NN(j)$ which contains all non-neighbors of v_j in G. Although node v_j is not adjacent to itself v_j is not considered to be an element of $NN(j)$. The cardinality of $NN(j)$ is denoted by h_j. More preciselly

$$h_j = \begin{cases} |NN(j)|, \text{ if } NN(j) \neq \emptyset \\ 1, \qquad\quad \text{if } NN(j) = \emptyset \end{cases}$$

or equivalently we set $h_j = \max\{1, |NN(j)|\}$. Since we are assuming that G does not contain any full rank node we may afford to be a little sloppy. Croce and Tadei (1994) have advanced the following linear program

$$\begin{aligned} &x_1 + \cdots + x_n \to \max \\ &h_j x_j + \textstyle\sum_{i \in NN(j)} x_i \leq h_j, \text{ for each } j, 1 \leq j \leq n \end{aligned}$$

to solve the maximum clique problem.

Let us suppose that the nodes of the graph G are listed in the way $v_1, v_2, \ldots, v_n$ and we keep this ordering of the nodes fixed. Let $NN^+(i)$ be the set of non-neighbors of the node v_j in the set $\{v_{j+1}, \ldots, v_n\}$ and we set

$$h_j^+ = \begin{cases} |NN^+(j)|, & \text{if } NN^+(j) \neq \emptyset \\ 1, & \text{if } NN^+(j) = \emptyset \end{cases}$$

or equivalently we set $h_j^+ = \max\{1, |NN^+(j)|\}$. The reader can verify that the linear program

$$\begin{aligned} & x_1 + \cdots + x_n \to \max \\ & h_j^+ x_j + \textstyle\sum_{i \in NN^+(j)} x_i \leq h_j^+, \text{ for each } j, 1 \leq j \leq n \end{aligned}$$

can be used to solve the maximum clique problem . We may call this LP the triangle shape formulation of the maximum clique problem. The reason we included this reformulation is that certain LP solvers work very rapidly with triangle shape constraint matrix.

The number of the variables is n in each program, where n is the number of nodes of the given graph G. The number of the constraints is n in the Croce–Tadei and in the triangle shape reformulations. The number of constraints is $O(n^2)$ in the edge reformulation. The number of constraints in the independent set reformulation can be $O(2^n)$. There are graphs having a large number of maximal independent sets. The Bron–Kerbosch algorithm (Bron and Kerbosch 1973) can be used to generate all maximal cliques of the input graph. Applying the Bron–Kerbosch algorithm to the complement of G the maximal independent sets of G will be available. Thus there is a practical way to set up the independent set reformulation when the number of the nodes of the graph G is not overly large.

Listing all maximal cliques in order to find the a maximum clique does not look a sensible idea at the first glance. When the edge density of the graph G is low, that is, when the graph is sparse the maximum clique problem is not too hard. We will use the linear program only when the graph G is dense. In this situation $\overline{G}$ the complement of the graph G is sparse and the problem of listing all maximal cliques of $\overline{G}$ can be much easier than locating a maximum clique in G.

We will call the node v_i of the graph a full degree node of G if v_i is adjacent to each other node of the graph G. Clearly a full degree node v_i has degree $n-1$ and the problem of finding a maximum clique in G can be reduced to the problem of finding a maximum clique in the graph induced by $V \setminus \{v_i\}$. When v_i is a full degree node in G, then the variable x_i is missing from the constraints of the formulations we described. But x_i is present in the objective function. Therefore $x_i = 1$ must hold in each optimal solution. It is straight-forward to detect full degree nodes in a graph. From this reason we assume that we deal with the situation when the given graph G does not have any full degree node.

3 Connections between the LP reformulations

There are intimate connections among the three reformulations. Let I be an independent set of G with three elements. For the sake of definiteness suppose that $I = \{v_1, v_2, v_3\}$. The inequalities $x_1 + x_2 \leq 1$, $x_2 + x_3 \leq 1$, $x_1 + x_3 \leq 1$ are constraints of the edge reformulation. Adding them up gives $2x_1 + 2x_2 + 2x_3 \leq 3$. Dividing by 2 we get $x_1 + x_2 + x_3 \leq 1.5$. Since the left hand side is an integer we may chop off the fractional part of the right hand side which gives $x_1 + x_2 + x_3 \leq 1$. This inequality is a constraint of the independent set reformulation. In short a constraint in the independent set reformulation which is associated with an independent set of cardinality three belongs to the rank 1 Chvátal closure of the edge reformulation (Chvátal 1973).

Let I be an independent set of G with four elements. For the sake of simplicity assume that $I = \{v_1, v_2, v_3, v_4\}$. The inequalities $x_1+x_2 \leq 1, x_1+x_3 \leq 1, x_1+x_4 \leq 1$, $x_2 + x_3 \leq 1$, $x_2 + x_4 \leq 1$, $x_3 + x_4 \leq 1$ are constraints of the edge reformulation. As we have seen the inequalities $x_1 + x_2 + x_3 \leq 1$, $x_1 + x_2 + x_4 \leq 1$, $x_1 + x_3 + x_4 \leq 1$, $x_2 + x_3 + x_4 \leq 1$ belong to the rank 1 Chvátal closure of the edge reformulation. Adding them up gives $3x_1 + 3x_2 + 3x_3 + 3x_4 \leq 4$. Dividing by three and rounding on the right hand side leads to $x_1 + x_2 + x_3 + x_4 \leq 1$. Thus a constraint in the independent set reformulation that is associated with an independent set of cardinality four belongs to the rank 2 Chvátal closure of the edge reformulation. In general if I is an independent set of G with $|I| = s$, then the constraint in the independent set reformulation associated with I belongs to the rank $(s - 2)$ Chvátal closure of the edge reformulation.

Since the constraints of the independent set reformulation are in the Chvátal closure of the edge reformulation (with various ranks) the independent set reformulation is a tighter formulation of the maximum clique problem than the edge reformulation. (For more details about the Chvátal rank of a constraint see Chvátal (1973).)

Let us consider the edge reformulation. We may collect all constraints containing variable x_j. Adding up these constraints we get the j-th constraint of the Croce–Tadei reformulation. In fact, for 0–1 variables these constraints are equivalent, as they both proper formulations.

3.1 Case of continuous variables

If instead of using 0–1 variables we use continuous variables, we look for the solution of the relaxed problem (Dantzig 1993), the Croce–Tadei reformulation is only a consequence of the edge reformulation, and the latter is tighter. This can be proven by a simple example. Consider the path P_4, consisting of nodes 1, 2, 3, 4, which are represented in the LP by variables x_1, x_2, x_3, x_4. The constraints for the edge reformulation are:

$$\begin{aligned} x_1 + x_3 &\leq 1 \\ x_1 + x_4 &\leq 1 \\ x_2 + x_4 &\leq 1 \end{aligned}$$

The constraints for the Croce–Tadei reformulation are:

$$\begin{aligned} 2x_1 + x_3 + x_4 &\leq 2 \\ 1x_2 + x_4 &\leq 1 \\ 1x_3 + x_1 &\leq 1 \\ 2x_4 + x_1 + x_2 &\leq 2 \end{aligned}$$

The substituted values $x_1 = 2/3, x_2 = 0, x_3 = 0, x_4 = 2/3$ forms a feasible solution of the Croce–Tadei reformulation but not a solution of the edge reformulation. In general, the set of feasible solutions of the continuous relaxation of the Croce–Tadei reformulation can be strictly larger than the set of the feasible solutions of the edge reformulation.

On the other hand, even if the reformulation is less tight, the Croce–Tadei reformulation uses much less number of constrains. It may be the case that the linear program for the Croce–Tadei reformulation can fit into the memory of the computer. So on one hand the edge reformulation may give a better upper estimate, but sometimes it is not solvable as being too big for computer memory.

4 Tightening the formulations

As we have seen all reformulations of the clique problem are equivalent in 0–1 LP, as the sets of the solutions are the same. In continuous LP that is for the relaxed problem they may and are differ. This fact is important for two reasons. First, sometimes one would only seek for an upper bound by solving the relaxed problem. Second, the solvers for 0–1 LP mostly use some Branch-and-Bound method and solve several times the relaxed problem to find the integral solution. In both cases tightening the formulation has crucial role.

Let $G = (V, E)$ be a finite simple graph such that $|V| = n$ and G does not have any full degree node.

Let $\{x_1, \ldots, x_n\} \in [0, 1]$: note that $x_1 = 0.5, \ldots, x_n = 0.5$ is a feasible solution of the edge and the Croce–Tadei formulations. This means that $n/2$ is the lower bound for the objective function of the relaxed problem for both reformulations. Thus for those cases, where $\omega(G) \ll n/2$ the relaxed optimum is not a good upper bound. Note, that for hard clique problems this is usually the case, as for cases if $\omega(G)$ is near to n, that is $n - \omega(G)$ is small, the problem coincides with the vertex cover problem and can be solved in fixed parameter time (Cygan et al. 2015).

Our first proposed method is a tactical modification of the Croce–Tadei formulations. Let H_i be the subgraph of G induced by $NN(i)$ and set $\alpha_i = \omega(H_i)$. The reader may notice that the number h_i can be replaced by α_i in the Croce–Tadei formulation for each i, $1 \leq i \leq n$. When $\alpha_i < h_i$ the new formulation is tighter than the original. For example $x_1 = 0.5, \ldots, x_n = 0.5$ is not a feasible solution any longer. In case the graphs $H_1, \ldots, H_n$ are too large to compute the clique numbers $\omega(H_1), \ldots, \omega(H_n)$, then we may use any upper bounds of these numbers we may lay our hands on. For example if the nodes of H_i can be legally colored using β_i colors then $\alpha_i \leq \beta_i$ holds and consequently h_i can be replaced by β_i.

Our proposed second method for tightening the formulation is by adding new constraints. Suppose that I is an independent set of G such that $|I| \geq 3$. For the sake of definiteness suppose that $I = \{v_1, v_2, v_3, v_4\}$. Now the constraint $x_1+x_2+x_3+x_4 \leq 1$ can be appended to the edge and the Croce–Tadei formulations. In this way we get tighter formulations. For example $x_1 = 0.5, \ldots, x_n = 0.5$ is not a feasible solution any longer.

We have carried out a large scale numerical experiment to compare these formulations. The results are summarized in tables. The details are described in Sect. 7.

5 Generating new cuts

Note that legal coloring of the nodes can be used to construct independent sets of the given graph G. In fact, a color class C of a legal coloring of the nodes is an independent set. The constraint $\sum_{i \in C} x_i \leq 1$ then can be appended to the LP formulation of the maximum clique problem. It may happen that this new constraint sorts out the optimal solution of the continuous version of the LP and we get a better upper estimate of the clique number of the graph G.

There are further ways to locate independent sets in the given graph G. For instance applying clique search algorithm to the complement graph $\overline{G}$ can be used to find (not necessarily optimal) cliques in $\overline{G}$ which in turn can provide independent sets in G.

We generalize the concept of independent set. Let $G = (V, E)$ be a finite simple graph and let s be an integer. A set I of V is called an s-clique free set if the subgraph of G induced by I does not contain any s-clique. (See Szabó 2011; Szabó and Zaválnij 2012.) The independent sets of G are the 2-clique free sets of G. Note that if I is an s-clique free set in G, then the inequality $\sum_{i \in I} x_i \leq s - 1$ must hold.

Any maximum clique algorithm can be used to locate various s-clique free sets. Let I be a subset of V and let H be the subgraph of G induced by I. If $\omega(H) = k$, then I is a $(k + 1)$-clique free set of G. In our computations we used the Östergård algorithm (Östergård 2002) to locate s-clique free sets.

Here is what we have done. We started with a 0–1 LP formulation of the maximum clique problem. We have solved the continuous relaxation of the LP and get the optimal solution $[\alpha_1, \ldots, \alpha_n]$. We have rearranged the components of the optimal solution to a decreasing order to get $\gamma_1, \ldots, \gamma_n$. here γ_1 is the largest and γ_n is the smallest among the numbers $\alpha_1, \ldots, \alpha_n$. There is a permutation $p(1), \ldots, p(n)$ of the elements $1, \ldots, n$ such that $\gamma_j = \alpha_{p(j)}$ for each j, $1 \leq j \leq n$. We consider the sets of nodes $I_j = \{v_{p(1)}, \ldots, v_{p(j)}\}$. If I_j is an s-clique free set of G and $s-1 < \alpha_{p(1)}+\cdots+\alpha_{p(j)}$, then the optimal solution $[\alpha_1, \ldots, \alpha_n]$ violates the constraint $x_{p(1)}+\cdots+x_{p(j)} \leq s-1$. In short, using Östergård algorithm we may find a cut for the LP we are working with. This is a more systematic way to construct tighter LP formulation. It may happen that the graph induced by the set I_j is too large for the combinatorial maximum clique algorithm. In this case we give up our attempt to tighten the formulation in this way. Also it may happen adding a large number of new cuts constructed in this way do not improve the upper estimate of the clique sufficiently. In this case again we abandon the attempt to tighten the formulation. This is the time to divide the clique search into smaller instances.

6 Mixed integer programming approach

In previous sections we described both 0–1 LP and continuous (relaxed) LP solutions of the clique problem. The first approach is exact but obviously slow, while the second approach gives us only an upper bound but much faster. We would like to spend more time to produce a better upper bound. The Mixed Integer Programming (MIP) approach can provide this. If one would prescribe some of the variables to be binary and the other variables to be continuous then the optimum value probably go down (never up) and we would get a better upper bound. The question is which and how many of the variables should we prescribe to be binary?

It is an empirical fact that the running time of a mixed integer program is greatly influenced by the number of the integer variables and less sensitive to the number of the continuous variables. In our MIP approach with the LP reformulation of the maximum clique problem we present here most of the variables should be continuous and a few variables should be integer. In this section we describe some MIP reformulation of maximum clique problem. The Tables 1, 2, 3 and 4 show ILP results, while Table 5 shows MIP results.

Let $[\alpha_1, \ldots, \alpha_n]$ be the optimal solution of the continuous relaxation of the 0–1 LP formulation. Our assumption was when we solve the relaxation of LP, the variables which will be close to 1 may be members of the maximum clique. Thus we choose the average value as cut-off, and so variables over average to be binary and below average to be continuous. Namely, we computed $\overline{\alpha}$ the average of the α_j values. We kept the variable x_j continuous whenever the inequality $\overline{\alpha} > \alpha_j$ holds and made the remaining variables integer. Note, that this approach can be iterated, and new variables can be chosen to be binary from the continuous variables by looking at the solution of the MIP. We continue this until we got solution in reasonable time or all variable reach 0 or 1.

The steps for the described algorithm are:

1. solve the MIP and examine the value of the MIP variables;
2. set a part of variables to integer by using heuristic described above;
3. after prescribing variables to be binary repeat from step 1.

The objective value of the MIP solutions getting closer to the maximum clique size with each iteration. Naturally this approach is not particularly promising when the $\alpha_1, \ldots, \alpha_n$ numbers are all equal or are close to each other.

7 The numerical results

In this section we describe the test problems we used for our experiments and we present the results of the numerical experiments we carried out. We used test problems from various sources to compare the canonical and our new formulations. The graphs are the most commonly used DIMACS benchmark test problems from the second DIMACS challenge[1] (Hasselberg et al. 1993), graphs of combinatorial problems of

[1] ftp://dimacs.rutgers.edu/pub/challenge/.

monotonic matrices (Weisstein, Szabó 2013), and well-known graphs coming from coding theory, namely Deletion-Correcting Codes[2] (Sloane).

7.1 Canonical formulations

In Table 1 we collected the relaxed optimum values of several canonical formulations, which gives us an upper bound for the clique size. The running time for the LP solver was always below 1 minute. The first column ("graph") of Table 1 holds the name of the graph. The second, third, fourth columns contain the number of nodes ("$|V|$"), then number of edges ("$|E|$"), and the clique number of the graph ("ω"), respectively.

The Bron–Kerbosch algorithm is designed to list all maximal cliques of a given graph G. The maximal cliques of G are independent sets of the complement graph $\overline{G}$ and so the Bron–Kerbosch algorithm can be used to set up the independent set 0–1 LP formulation of the maximum clique problem. The continuous relaxation of the 0–1 program provides an upper bound for $\omega(G)$. In Table 1 the column labeled by $\overline{\omega}_{BK}$ contains these estimates.

Since listing all maximal independent sets of a given graph is computationally demanding we have experimented with relaxation of the independent set formulation. Namely, instead of using all independent sets we used only a few of them which are relatively easy to compute.

Choosing a node v with maximum degree in the graph G and restricting G to the neighbors of v then repeating this procedure in connection with the remaining graph eventually leads to a not necessarily maximum clique in G. We refer to this method as the maximum degree rule. Applying the maximum degree rule to the complement graph $\overline{G}$ provides an independent set in G. After deleting the located independent set we may locate a new independent set in the remaining graph. The family of independent sets we collect in this way are used to set up a relaxed version of the independent set formulation. In Table 1 the column labeled by $\overline{\omega}_M$ contains the upper bound for $\omega(G)$ we obtain in this way.

In place of the maximum degree rule one can use, say the Östergård algorithm, which locates a maximum clique not only a suboptimal clique. In Table 1 the column labeled by $\overline{\omega}_O$ contains the upper bound for $\omega(G)$ one can get in this manner.

If each node of the graph G is colored with exactly one color such that adjacent nodes do not receive the same color, then we say that the nodes of G are legally colored. The set of nodes receiving the same color is called a color class. Plainly color classes of a legal coloring of the nodes of G are independent sets of G. For a finite simple graph G there is an integer k such that the nodes of G can be colored with k colors legally and cannot be colored with $(k-1)$ colors legally. This well defined number k is called the chromatic number of G and it is denoted $\chi(G)$. Determining $\chi(G)$ is an NP-hard problem. However greedy algorithms can provide legal colorings with suboptimal number of colors relatively easily. So we may add the constraints associated with the color classes to any LP formulation of the maximum clique problem. In Table 1 the columns labeled by $\overline{\omega}_{MC}, \overline{\omega}_{OC}$ contain the upper bound for $\omega(G)$ we get when added

[2] http://neilsloane.com/doc/graphs.html.

Table 1 Upper bounds obtained by relaxation of the canonical formulations and added new cuts

Graph	$\|V\|$	$\|E\|$	ω	$\overline{\omega}_M$	$\overline{\omega}_{MC}$	$\overline{\omega}_O$	$\overline{\omega}_{OC}$	$\overline{\omega}_{BK}$
brock200_3	200	12,048	15	33.63	33.10	31.75	31.30	**27.23**
brock200_4	200	13,089	17	36.49	35.49	35.18	**34.74**	–
brock400_2	400	59,786	29	74.02	73.07	–	–	**64.27**
c-fat500-5	500	23,191	64	**64.00**	**64.00**	–	–	–
c-fat500-10	500	46,627	126	**126.00**	**126.00**	–	–	–
hamming10-2	1024	518,656	512	**512.00**	**512.00**	**512.00**	**512.00**	**512.00**
hamming10-4	1024	434,176	40	89.37	**74.00**	–	–	–
keller4	171	9435	11	15.00	15.00	15.00	15.00	**14.82**
keller5	776	225,990	27	**31.00**	**31.00**	–	–	–
p_hat300-1	300	10,933	8	27.28	**22.00**	–	–	–
p_hat300-2	300	21,928	25	50.23	**42.89**	–	–	–
p_hat300-3	300	33,390	36	67.06	64.14	64.23	62.14	**54.31**
p_hat500-3	500	93,800	50	105.42	100.85	–	–	**83.20**
san200_0.7_1	200	13,930	30	35.21	**30.00**	35.31	**30.00**	**30.00**
san200_0.7_2	200	13,930	18	30.34	19.00	21.86	**18.00**	**18.00**
san200_0.9_3	200	17,910	44	58.42	**44.00**	50.03	**44.00**	**44.00**
monoton-7	343	46,305	19	33.00	28.00	30.50	27.66	**23.97**
monoton-8	512	106,624	23	43.00	40.00	32.00	32.00	**30.89**
matr-del-8	256	28,801	30	42.24	38.88	38.97	36.85	**30.00**
matr-del-9	512	121,089	52	75.56	71.27	69.04	67.33	**53.31**

the color class constraint to the formulations coming from the maximum degree rule and the Östergård algorithm.

Table 1 shows the best result in bold face. As one would expect the Bron–Kerbosch algorithm, which lists all maximal independent sets, gives the best result in the most cases, but when the graph is sparse the complementary is dense and the algorithm cannot finish in reasonible time (for example c-fat500-5, c-fat500-10 in the table have "−″, because the graph is sparse so the algorithm cannot not finished within 2 hours). There are some cases when Östergård algorithm with color classes can be successfully completed but Bron–Kerbosch cannot. Other cases, when none of them can run, we could use maximum degree rule with color classes to obtain the best result. Notable result is that we could obtain optimal upper bound in 7 out of 20 cases.

It is a widely held opinion that tighter formulation leads to sharper upper bound, that is the objective value of the relaxed problem, for the clique number of the given graph, which in turns translates to shorter running time for solving the ILP. To check this we produced Table 2, which contains the running times of ILP solutions in seconds. The text "> 10" appears when we interrupted the program running after 10 hours. "−″ show us if the necessary precondition could not run (create independent sets, color classes etc.) The results does not meet the expectation and there seems no connection between the relaxed objective value and ILP running times.

Table 2 The running times in seconds of ILP solution for previous formulations

Graph	$\|V\|$	$\|E\|$	ω	$\overline{\omega}_M$	$\overline{\omega}_{MC}$	$\overline{\omega}_O$	$\overline{\omega}_{OC}$	$\overline{\omega}_{BK}$
brock200_3	200	12,048	15	> 10h	731	> 10h	> 10h	7548
brock200_4	200	13,089	17	> 10h	2880	> 10h	> 10h	–
brock400_2	400	59,786	29	> 10h	> 10h	–	–	> 10h
c-fat500-5	500	23,191	64	5	5	–	–	–
c-fat500-10	500	46,627	126	5	8	–	–	–
hamming10-2	1024	518,656	512	0	0	0	0	2
hamming10-4	1024	434,176	40	> 10h	> 10h	–	–	–
keller4	171	9435	11	341	281	397	228	544
keller5	776	225,990	27	> 10h	> 10h	–	–	–
p_hat300-1	300	10,933	8	> 10h	354	–	–	–
p_hat300-2	300	21,928	25	> 10h	> 10h	–	–	–
p_hat300-3	300	33,390	36	> 10h	> 10h	> 10h	> 10h	> 10h
p_hat500-3	500	93,800	50	> 10h	> 10h	–	–	> 10h
san200_0.7_1	200	13,930	30	13	0	8	1	0
san200_0.7_2	200	13,930	18	> 10h	34	113	58	5
san200_0.9_3	200	17,910	44	> 10h	221	169	119	13
monoton-7	343	46,305	19	> 10h	> 10h	> 10h	> 10h	18025
monoton-8	512	106,624	23	> 10h	> 10h	> 10h	> 10h	> 10h
matr-del-8	256	28,801	30	> 10h	> 10h	> 10h	> 10h	0
matr-del-9	512	121,089	52	> 10h	> 10h	> 10h	> 10h	15637

7.2 Croce–Tadei formulation and its tightening

Results of the numerical experiments in connection with the relaxed Croce–Tadei reformulation are presented in Table 3 indicating the best result in bold face. Column headed by $\widehat{\omega}$ contains the estimate of $\omega(G)$ provided by the continuous relaxation of the original Croce–Tadei 0–1 LP formulation. The estimate provided by the tighter formulation in which h_j the cardinality of the set $NN(j)$ is replaced by α_j the clique number of the graph H_j induced by $NN(j)$ is in the column labeled by $\widehat{\omega}_O$. (We used the Östergård algorithm to compute the clique number.)

Legal coloring of the nodes of G is referred as global coloring while the legal coloring of the nodes of H_j is referred as local coloring. The column headed by $\widehat{\omega}_{OC}$ contains the estimates of the tighter version when we added constraints associated with the color classes of a greedy global coloring.

In the next numerical experiment we did not compute the clique number α_j of the graph H_j rather we simply used β_j the number of colors coming from a greedy legal coloring of the nodes of H_j. These estimates are listed in the columns labeled by $\widehat{\omega}_L$. Adding constraints associated with the colors classes of a greedy global coloring provides the estimates in the column marked by $\widehat{\omega}_{LC}$.

We carried out similar experiments in connection with the triangle shape formulation. Also we combined the constraints of each of the reformulations we have listed.

Table 3 Upper bound obtained by relaxation of the Croce–Tadei formulation and some of its tightened versions

Graph	$\|V\|$	$\|E\|$	ω	$\widehat{\omega}$	$\widehat{\omega}_O$	$\widehat{\omega}_{OC}$	$\widehat{\omega}_L$	$\widehat{\omega}_{LC}$
brock200_3	200	12,048	15	100.0	24.82	**24.80**	39.93	36.00
brock200_4	200	13,089	17	100.0	29.90	**29.77**	44.35	37.00
brock400_2	400	59,786	29	200.0	–	–	94.43	**80.00**
c-fat500-5	500	23,191	64	250.0	67.93	**64.00**	67.93	**64.00**
c-fat500-10	500	46,627	126	250.0	143.46	**126.00**	143.46	**126.00**
hamming10-2	1024	518,656	512	512.0	**512.00**	**512.00**	**512.00**	**512.00**
hamming10-4	1024	434,176	40	512.0	–	–	124.89	**74.00**
keller4	171	9435	11	85.5	21.12	**15.00**	28.11	**15.00**
keller5	776	225,990	27	388.0	–	–	94.16	**32.00**
p_hat300-1	300	10,933	8	150.0	9.83	**9.82**	22.74	22.00
p_hat300-2	300	21,928	25	150.0	34.49	**33.94**	49.60	42.96
p_hat300-3	300	33,390	36	150.0	–	–	78.22	**65.96**
p_hat500-3	500	93,800	50	250.0	–	–	123.91	**103.00**
san200_0.7_1	200	13,930	30	100.0	–	–	41.51	**30.00**
san200_0.7_2	200	13,930	18	100.0	–	–	29.31	**19.00**
san200_0.9_3	200	17,910	44	100.0	79.17	**44.00**	79.28	**44.00**
monoton-7	343	46,305	19	171.5	28.09	**27.91**	30.86	28.00
monoton-8	512	106,624	23	256.0	37.20	**37.14**	41.45	40.00
matr-del-8	256	28,801	30	128.0	33.81	**33.70**	34.70	34.51
matr-del-9	512	121,089	52	256.0	59.94	**59.57**	61.16	60.67

The results were less good as for the original Croce–Tadei formulation, so in order to keep the presentation short we decided not to include all of our measurements.

Croce–Tadei formulation with Östergård algorithm with color classes shows the best result in accuracy. Other cases, when Östergård algorithm cannot give result us in polynomial time, legal coloring could help us with color classes. Östergård algorithm has the same problem like Bron–Kerbosch algorithm: when the graph has low density the algorithm cannot run winthin the foreseeable future.

If we take a look at Tables 1 and 3, we can see that heuristics improved the estimation of the maximum clique number compared to original canonical formulations with exception of the case when we list all maximal independent sets. The second set concluded optimal with 5 from 20 cases. Again, we checked if there is a connection between the objective value of the relaxed problem and the ILP solution running times. This time as shown in Table 4 there is a clear connection, and the presumption that tighter upper bound leads to faster ILP solution times turned out to be true for all cases. This matter needs to be investigated more thoroughly in the future.

7.3 Generating *s*-clique free cuts

In our next numerical experiment we investigate effect of the s-clique free cuts described in Sect. 5. We used the predefined ordering of the nodes of the underly-

Table 4 The running time in seconds for ILP solution of Croce–Tadei formulation and some of its tightened versions

Graph	$\|V\|$	$\|E\|$	ω	$\widehat{\omega}$	$\widehat{\omega}_O$	$\widehat{\omega}_{OC}$	$\widehat{\omega}_L$	$\widehat{\omega}_{LC}$
brock200_3	200	12,048	15	> 10h	4928	3093	> 10h	> 10h
brock200_4	200	13,089	17	> 10h	> 10h	19848	> 10h	> 10h
brock400_2	400	59,786	29	> 10h	–	–	> 10h	> 10h
c-fat500-5	500	23,191	64	> 10h	168	0	124	0
c-fat500-10	500	46,627	126	> 10h	> 10h	0	> 10h	0
hamming10-2	1024	518,656	512	0	0	0	0	0
hamming10-4	1024	434,176	40	> 10h	–	–	> 10h	> 10h
keller4	171	9435	11	> 10h	19091	416	> 10h	248
keller5	776	225,990	27	> 10h	–	–	> 10h	> 10h
p_hat300-1	300	10,933	8	> 10h	41	43	> 10h	> 10h
p_hat300-2	300	21,928	25	> 10h	> 10h	> 10h	> 10h	> 10h
p_hat300-3	300	33,390	36	> 10h	–	–	> 10h	> 10h
p_hat500-3	500	93,800	50	> 10h	–	–	> 10h	> 10h
san200_0.7_1	200	13,930	30	> 10h	–	–	> 10h	0
san200_0.7_2	200	13,930	18	> 10h	–	–	> 10h	6
san200_0.9_3	200	17,910	44	> 10h	> 10h	15	> 10h	12
monoton-7	343	46,305	19	> 10h	> 10h	> 10h	> 10h	> 10h
monoton-8	512	106,624	23	> 10h	> 10h	> 10h	> 10h	> 10h
matr-del-8	256	28,801	30	> 10h	> 10h	> 10h	> 10h	> 10h
matr-del-9	512	121,089	52	> 10h	> 10h	> 10h	> 10h	> 10h

ing graph and performed the Östergård algorithm for 10 s. The new s-clique free cut was added to the LP formulation. This procedure was iteratively repeated. Table 5 shows that the s-clique free cuts in the continuous relaxation solution are improving the upper estimate for $\omega(G)$. Though at the beginning the improvement is substantial as we add more and more cuts the improvement diminishes. In the end if we would like to reduce the optimum value we need to add too many constrains and so the running time of solving the problem increases. The results in the table are connected to the graph p_hat300-2. This behavior is typical and so we decided not to include further similar measurements. Note, that this approach could reduce the objective function value as well as shorten the running time. Though further experiments are needed to tune the number of cuts needed to be added for optimal performance.

7.4 Iterated MIP approach

The last numerical experiment we performed was the iterated approach for mixed integer reformulation of the maximum clique problem. We performed the procedure described in Sect. 6 and set the above average variables to be binary. In the next iteration we solved the MIP problem, and again set the above average real variables to binary. If a variable was set to binary it remains so. We proceeded with the iterations

Table 5 Number of s-clique free cuts and the obtained upper estimate for the p_hat300-2 graph

Number of cuts	Estimate	Running time (s)
21	30.44	1827
40	30.02	1747
57	29.99	1688
74	29.85	1898
407	28.99	3340

Table 6 Number of binary variables needed for solving the Mixed Integer Programming formulation to optimality

Graph	$\|V\|$	$\|E\|$	$\omega(G)$	0–1 Variables	Running time (s)
monoton-5	125	5500	11	47	2
monoton-6	216	17,550	14	80	501
C125.9	125	6963	34	53	14
gen400_p0.9_55	400	71,820	55	170	11 033
MANN_a27	378	70,551	126	27	2
MANN_a45	1035	533,115	345	45	290

till all binary and real variables reached 0 or 1 value. For some graphs this procedure could end in time limit thus producing optimal value, and for some graphs it could not. Earlier mentioned methods with ILP do not terminate in a given time limit for some cases shown in Tables 2 and 4. This approach when proceed, gives us the optimal solution in our experiments. The last column of Table 6 contains the number of integer variables needed for reaching optimality in such mixed integer program. Naturally this approach is not particularly promising when the $\alpha_1, \ldots, \alpha_n$ numbers are all equal or are close to each other.

Acknowledgements Dóra Kardos and Patrik Patassy have been supported by the European Union, cofinanced by the European Social Fund (EFOP-3.6.3-VEKOP16-2017-00002). The research of Sándor Szabó and Bogdán Zaválnij was supported by National Research, Development and Innovation Office – NKFIH Fund No. SNN–135643.

Funding Open access funding provided by ELKH Alfréd Rényi Institute of Mathematics.

References

Beke Á, Szabó S, Zaválnij B (2021) Some zero-one linear programming reformulations for the maximum clique problem. Mathematica Pannonica 27(1):32–47

Bomze IM, Budinich M, Pardalos PM, Pelillo M (1999) The maximum clique problem, handbook of combinatorial optimization, vol 4, Kluwer Academic Publisher

Bron C, Kerbosch J (1973) Finding all cliques of an undirected graph. Commun ACM 16:575–577

Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Oper Res Lett 9:375–382

Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math 4:305–337

Croce FD, Tadei R (1994) A multi-KP modeling for the maximum-clique problem. Eur J Oper Res 73:555–561

Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Saurabh S (2015) Parameterized algorithms. Springer

Dantzig GB (1993) Linear programming and extensions. Princeton University Press, Princeton

Garey MR, Johnson DS (2003) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York

Hasselberg J, Pardalos PM, Vairaktarakis G (1993) Test case generators and computational results for the maximum clique problem. J Glob Optim 3:463–482

Konc J, Janežič D (2007) An improved branch and bound algorithm for the maximum clique problem. MATCH Commun Math Comput Chem 58:569–590

Kumlander D (2005) Some practical algorithms to solve the maximal clique problem, PhD. Thesis. Tallin University of Technology

Östergård PRJ (2002) A fast algorithm for the maximum clique problem. Discrete Appl Math 120:197–207

Papadimitriou CH (1994) Computational complexity. Addison-Wesley Publishing Company Inc, Reading

Sloane NJA, Challenge problems: independent sets in graphs. http://neilsloane.com/doc/graphs.html

Szabó S (2011) Parallel algorithms for finding cliques in a graph. J Phys Conf Ser 268:012030. https://doi.org/10.1088/1742-6596/268/1/012030

Szabó S, Zaválnij B (2012) Greedy algorithms for triangle free coloring. AKCE Int J Graphs Comb 9(2):169–186

Szabó S (2013) Monotonic matrices and clique search in graphs. Ann Univ Sci Budapest Sect Comput 41:307–322

Tomita E, Seki T (2003) An efficient branch-and-bound algorithm for finding a maximum clique. Lect Not Comput Sci 2631:278–289

Weisstein EW, Monotonic matrix. In: MathWorld: a wolfram web resource. http://mathworld.wolfram.com/MonotonicMatrix.html