Testing Copositivity via Mixed–Integer Linear Programming

Kurt M. Anstreicher*

March 3, 2020

Abstract

We describe a simple method to test if a given matrix is copositive by solving a single mixed-integer linear programming (MILP) problem. This methodology requires no special coding to implement and takes advantage of the computational power of modern MILP solvers. Numerical experiments demonstrate that the method is robust and efficient.

Keywords: Copositive Matrix, Copositive Programming, Mixed-Integer Linear Programming.

1 Introduction

An $n \times n$ symmetric matrix A is called *copositive* if $x^T Ax \geq 0$ for all $x \in \mathbb{R}^n_+$. The copositive matrices of dimension n form a closed convex cone, which we denote by $\mathcal{COP}^n$. The dual of $\mathcal{COP}^n$, which we denote $\mathcal{CP}^n$, is the cone of *completely positive* matrices; $n \times n$ matrices which can be written in the form $\sum_{i=1}^k u_i u_i^T$, where each $u_i \in \mathbb{R}^n_+$. Copositive matrices have long been studied in the linear algebra literature; see for example the survey article [13] and references therein. In recent years there has been growing interest in copositive matrices among researchers in optimization due to the fact that some NP-hard optimization problems can be posed as linear problems over $\mathcal{COP}^n$, or alternatively $\mathcal{CP}^n$; see for example [2, 5, 7, 11].

A fundamental problem in the theory of copositive matrices is determining if $A \in \mathcal{COP}^n$ for a given matrix A. This recognition problem is known to be co-NP complete [10, 15] and therefore cannot be expected to be tractable in general. Nevertheless it is very desirable to devise

*Department of Business Analytics, University of Iowa, Iowa City, IA, 52242-1994, USA. Email: kurt-anstreicher@uiowa.edu.

methodologies for the recogntion problem that are as computationally efficient as possible. Algorithms that are capable of solving the general copositive recognition problem include methods based on simplicial decomposition [4, 18], polynomial programming [16] and finite branching for nonconvex quadratic programming [6, 20]. A recent paper by Dickinson [9] describes a simple methodology for recognizing copositivity based on earlier work in the linear algebra literature. Any method that solves the recognition problem can be expected to produce a vector $x \in \mathbb{R}^n_+$ with $x^T Ax < 0$ in the case that $A \notin \mathcal{COP}^n$, but a "short certificate" that $A \in \mathcal{COP}^n$ is more difficult to obtain. Of the algorithms described above, the methodology based on polynomial programming [16] appears to be the most capable of providing such a certificate of copositivity. The basic approach in [9] requires the solution of exponentially many linear systems to verify that $A \in \mathcal{COP}^n$, but [9] goes on to show that a more elaborate procedure can obtain a shorter certificate of copositivity in some cases.

The approach that we take in this paper is to adapt the algorithm for copositivity recognition from [9] so that it can be implemented by solving a single mixed-integer linear programming (MILP) problem. This has several advantages over the original method of [9]. First, no special algorithm coding is required; the required MILP has a very simple form and can easily be generated using any environment for mathematical programming. Second, the formulation as an MILP takes advantage of the fact that modern MILP solvers are extremely robust and efficient. For example, such codes automatically exploit the availability of multiple CPU cores and use advanced constraint-generation techniques to minimize branching. These added constraints can substantially reduce the amount of computation, especially in the case that $A \in \mathcal{COP}^n$.

In the next section we describe our MILP formulation for recognizing copositivity and prove its correctness. In Section 3 we describe numerical results using our MILP formulation. We consider families of test problems that generate both instances where $A \in \mathcal{COP}^n$ and where $A \notin \mathcal{COP}^n$. Computational results demonstrate that our method is robust and efficient, and to our knowledge appears to be the most computationally efficient algorithm for the general copositivity recognition problem to date.

Notation. We use $\mathcal{S}^n$ to denote $n \times n$ symmetric matrices, $\mathcal{S}^n_+$ to denote $n \times n$ symmetric positive semidefinite (PSD) matrices, and $\mathcal{N}^n$ to denote $n \times n$ symmetric non-negative matrices. The non-negative orthant in $\mathbb{R}^n$ is denoted $\mathbb{R}^n_+$, and $x \in \mathbb{R}^n_{++}$, or alternatively $x > 0$, denotes that $x_i > 0$, $i = 1, \ldots, n$. A vector with each component equal to one, of arbitrary dimension, is denoted by e, and $E = ee^T$. For $\beta \subset \{1, \ldots, n\}$, if $A \in \mathcal{S}^n$ then $A_{\beta\beta}$ is the principal submatrix of A with rows and columns indexed by the elements of β, and if $x \in \mathbb{R}^n$ then x_β is the vector in $\mathbb{R}^{|\beta|}$ with components indexed by β.

2 An MILP formulation for copositivity testing

Our approach for testing copositivity is based on the characterization in [9], which itself is based on the notion of an almost copositive matrix. An $n \times n$ matrix is called *almost copositive* if it is not copositive, and either $n = 1$ or every $(n-1) \times (n-1)$ principal submatrix is copositive. It is then known [12, 19] that A is almost copositive if and only if A is nonsingular and $A^{-1} \leq 0$. It is then easy to show that there are several other properties equivalent to almost copositivity, as described in the following theorem.

Theorem 1. [9] *Let $A \in \mathcal{S}^n$, $n \geq 1$. Then the following are equivalent:*

1. *A is almost copositive,*
2. *A is nonsingular and $-A^{-1} \geq 0$,*
3. *$\forall b \in \mathbb{R}^n_+ \ \exists x \in \mathbb{R}^n_+ \ Ax = -b$,*
4. *$\exists x \in \mathbb{R}^n_+ \ Ax = -e$,*
5. *$\exists x \in \mathbb{R}^n_+ \ Ax < 0$.*

Moreover, it is easy to see that if A is not copositive then it must have a principal submatrix that is almost copositive. One can therefore check if a matrix is copositive by enumerating over subsets $\beta \subset \{1, 2, \ldots, n\}$ and checking if $A_{\beta\beta} x_\beta = -e$ has a solution with $x_\beta \geq 0$; A is copositive if and only if no such subset exists. This is the basis for the procedure for checking copositivity in [9]. This procedure is exponential in the worst case, and in particular to verify that $A \in \mathcal{COP}^n$ one must check exponentially many subsets β.

The approach that we will use here is to formulate an MILP problem that attempts to find a principal submatrix of A that is almost copositive. Let a_i^T denote the ith row of A, and assume that $\theta \in \mathbb{Z}_+$, $m \in \mathbb{R}^n_{++}$. Consider the following MILP problem:

$$
\begin{aligned}
\text{MILP}_{\mathcal{COP}}: \quad \max \quad & \gamma \\
\text{s.t.} \quad & a_i^T x \leq -\gamma + m_i(1 - y_i), \ i = 1, \ldots, n \\
& \gamma \geq 0, \ 0 \leq x \leq y, \\
& y \in \{0,1\}^n, \ e^T y \geq \theta.
\end{aligned}
$$

Theorem 2. *Let $n \geq 1$, $\theta = 1$, $A \in \mathcal{S}^n$ and $m \in \mathbb{R}^n_{++}$. Then A is copositive iff the solution value in* $\text{MILP}_{\mathcal{COP}}$ *is zero. If in addition* $\text{diag}(A) \geq 0$ *then the same holds with $\theta = 2$.*

Proof. We will prove the equivalent statement that A is not copositive if and only if the solution value in $\text{MILP}_{\mathcal{COP}}$ is strictly positive. Assume first that $\theta \geq 1$ and the solution value in

MILP$_{\mathcal{COP}}$ is $\gamma^* > 0$. Let $\beta = \{i : y_i = 1\}$, $\alpha = \{1, 2, \ldots, n\} \setminus \beta$. From the constraints of MILP$_{\mathcal{COP}}$ we have $\beta \neq \emptyset$, $a_i^T x \leq -\gamma^*$, $i \in \beta$, where $x_\alpha = 0$, $x_\beta \geq 0$, $x_\beta \neq 0$. Therefore $x^T A x < 0$, so A is not copositive.

Next assume that A is not copositive, and $\theta = 1$. Then there is a principal submatrix $A_{\beta\beta}$ that is almost copositive, so $v_\beta = -A_{\beta\beta}^{-1} e \geq 0$, $v_\beta \neq 0$. Let $\mu = \max\{v_i : i \in \beta\}$, $\gamma = 1/\mu$, $y_i = 1$ for $i \in \beta$, $y_i = 0$ for $i \in \alpha$, $x_\beta = \gamma v_\beta$, $x_\alpha = 0$, where $\alpha = \{1, 2, \ldots, n\} \setminus \beta$. Then $0 \leq x \leq y$, $\gamma > 0$, and (x, y, γ) satisfy the constraints of MILP$_{\mathcal{COP}}$ for $i \in \beta$. The remaining constraints can be written $a_i^T x + \gamma \leq m_i$, $i \in \alpha$, and can all be satisfied by further multiplying (x, γ) by a sufficiently small positive scalar; note that such a scaling has no effect on the constraints for $i \in \beta$. The result is a solution (x, y, γ) that is feasible for MILP$_{\mathcal{COP}}$ with $\gamma > 0$, and therefore the solution value in MILP$_{\mathcal{COP}}$ is strictly positive. When diag$(A) \geq 0$, $A \notin \mathcal{COP}^n$ iff A has a principal submatrix of dimension greater than one that is almost copositive and we may assume that $\theta = 2$ in the above implication. □

Theorem 2 holds for any choice of $m > 0$, but obviously the choice of m will affect the value $\gamma^* > 0$ in the case where A is not copositive. Small values of m could result in $\gamma^* > 0$ that is numerically indistinguishable from zero, while the use of an excessively large m, such as $m = Me$ where M is a large positive number, could unnecessarily weaken the LP relaxation of MILP$_{\mathcal{COP}}$ and have a negative effect on solver performance. One reasonable choice for m can be based on the proof of Theorem 2, in particular the final scaling of (x, γ) that could be required to satisfy the constraints $i \in \alpha$ when A is not copositive. Using the values

$$m_i = 1 + \sum_{j \neq i} \{a_{ij} : a_{ij} > 0\} \tag{1}$$

ensures that this final scaling could only be required if $\gamma > 1$, in which case a rescaling by $1/\gamma$ results in a feasible solution of MILP$_{\mathcal{COP}}$ with objective value equal to 1.0.

3 Computational results

In this section we describe the results of applying MILP$_{\mathcal{COP}}$ to a variety of test problems, each corresponding to the choice of a matrix A. Instances of MILP$_{\mathcal{COP}}$ were solved using CPLEX 12.10.0 on a PC with an Intel i7-6700 quad-core CPU running at 3.40 GHz, with 16G of RAM and a 64-bit OS. CPLEX utilizes the quad-core architecture of the CPU to run up to eight concurrent threads. Unless otherwise stated, all problems solved correspond to MILP$_{\mathcal{COP}}$ using m from (1), with $\theta = 2$.

3.1 Small exceptional matrices

We first consider some small examples of "exceptional" copositive matrices. A matrix $A \in \mathcal{COP}^n$ is called exceptional if $A \notin \mathcal{SPN}^n = (\mathcal{DNN}^n)^*$ where $\mathcal{DNN}^n = \mathcal{S}^n_+ \cap \mathcal{N}^n$ is the cone of "doubly nonnegative" matrices, and $\mathcal{SPN}^n = \mathcal{S}^n_+ + \mathcal{N}^n$. Exceptional matrices are interesting from the standpoint of copositivity testing because membership in $\mathcal{SPN}^n$, unlike membership in $\mathcal{COP}^n$, can be established in polynomial time by solving an optimization problem posed over self-dual cones. The famous Horn matrix

$$\begin{pmatrix} 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 \end{pmatrix}$$

is an exceptional 5×5 matrix, and the Hoffman-Pereira matrix

$$\begin{pmatrix} 1 & -1 & 1 & 0 & 0 & 1 & -1 \\ -1 & 1 & -1 & 1 & 0 & 0 & 1 \\ 1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & 1 & -1 & 1 & -1 \\ -1 & 1 & 0 & 0 & 1 & -1 & 1 \end{pmatrix}$$

is an exceptional 7×7 matrix. If A is either the Horn matrix or the Hoffman-Pereira matrix, CPLEX solves $\text{MILP}_{\mathcal{COP}}$ in about 0.01 sec, using 50 nodes for the Horn matrix and 80 nodes for the Hoffman-Pereira matrix. In both cases the solution value is zero, since these matrices are both copositive. If one of the diagonal entries of the Horn matrix is decreased to 0.99, as in the computational results from [16], then CPLEX obtains a solution value of 0.005 using 57 nodes, and the solution x has $x^T Ax = -0.10$. Note that for small n, the number of MILP nodes may be of the same order, and can even exceed, 2^n, the number of subsets of $\{1, \ldots, n\}$, as would be required to enumerate the principal submatrices of A for the original algorithm of [9]. In an MILP code such as CPLEX, the complete branching tree with n binary variables has $2^{n+1} - 1$ nodes.

Table 1: Average solution time (sec) for perturbed $\mathcal{SPN}^n$ problems

n	$A(1)$	$A(2)$	$A(3)$	$A(4)$	$A(5)$
30	0.1215	0.1172	0.1157	0.1145	0.1276
40	0.1641	0.1591	0.1577	0.1528	0.1570
50	0.2401	0.2315	0.2386	0.2375	0.2479
60	0.3505	0.3539	0.3488	0.3808	0.4002
70	1.0615	1.0105	1.1226	1.0545	1.0766

Table 2: Average MILP nodes for solution of perturbed $\mathcal{SPN}^n$ problems

n	$A(1)$	$A(2)$	$A(3)$	$A(4)$	$A(5)$
30	3,635	3,651	3,389	3,497	3,334
40	7,339	7,351	7,057	7,218	7,255
50	15,069	15,692	15,356	14,723	16,138
60	27,718	27,623	26,797	29,531	31,547
70	100,130	95,950	102,060	95,730	102,680

3.2 Perturbed $\mathcal{SPN}$ matrices

We next consider computations based on perturbing an initial matrix in $\mathcal{SPN}^n$ to generate matrices that are not in $\mathcal{COP}^n$. The procedure that we use is as follows:

1. Generate a matrix $A \in \mathcal{S}^n$ with diagonal elements equal to one, and off-diagonal elements uniformly distributed in $[0, 1]$.

2. Generate a matrix $B \in \mathcal{S}^n$ with diagonal elements equal to zero, and off-diagonal elements uniformly distributed in $[0, 1]$.

3. Compute $\alpha_{\max} = \max\{\alpha : A - \alpha B \in \mathcal{SPN}^n\}$.

4. For $\epsilon > 0$ and $k = 1, \ldots K$ let $A(k) = A + (\alpha_{\max} + (k-2)\epsilon)B$, and solve MILP$_{\mathcal{COP}}$ using $A(k)$.

In Step 3, the value of $\alpha_{\max}$ is determined by solving a conic optimization problem using SeDuMi. In Tables 1 and 2 we give results on the average time and nodes required to solve instances of MILP$_{\mathcal{COP}}$ generated as described above, using 25 instances for each dimension $n = 30, 40, 50, 60, 70$. For these problems the value of $\alpha_{\max}$ is typically in the range $[0.4, 0.6]$, and our computations use $\epsilon = 0.001$. Due to the methodology for generating these problems

we know that $A(k) \in \mathcal{SPN}^n$ for $k = 1, 2$ and $A(k) \notin \mathcal{SPN}^n$ for $k = 3, 4, 5$. In a large majority of cases we find that $A(k) \notin \mathcal{COP}^n$ for $k = 3, 4, 5$ as well, but occasionally there is an instance where $A(k) \in \mathcal{COP}^n$ for $k > 2$. It is worth noting that $A(k) \notin \mathcal{SPN}^n$ could be established by solving a conic optimization problem similar to the problem that determines $\alpha_{\max}$, but the time required to solve such a problem using SeDuMi is much higher than the time to solve MILP$_{\mathcal{COP}}$ using CPLEX for the same matrix $A(k)$. In fact for the results reported in Tables 1 and 2, most of the computational time was spent determining the $\alpha_{\max}$ values. Looking at the results in the tables it is clear that the average MILP nodes required to solve these problems grows substantially for $n = 70$ compared to $n = 30$, but the average solution time is still only about 1.0 seconds for $n = 70$. In all cases the average number of MILP nodes is now a very small fraction of 2^n, the number of principal submatrices of A; 2^{30}=1.07E9 and 2^{70}=1.18E21. It is also clear that for a given n there is only minor variation in average time and nodes for the problems based on $A(k)$, $k = 1, \ldots, 5$.

3.3 Max-clique matrices

Our last computations are based on the well-know relationship [8, 14] between the maximum clique size in a graph and the copositivity of a certain matrix derived from the incidence matrix of the graph. In particular, if $\mathcal{I}_G$ is the incidence matrix for a graph G on n vertices, then the maximum clique size ω is given by

$$\omega = \min\{k \in \mathbb{Z}_+ : k(E - \mathcal{I}_G) - E \in \mathcal{COP}^n\}.$$

The maximum clique size can therefore be determined by checking if $A(k) = k(E - \mathcal{I}_G) - E \in \mathcal{COP}^n$ for varying values of k. For example, if G is the graph corresponding to vertices of the icosahedron [3], then $\omega = 3$, and solving MILP$_{\mathcal{COP}}$ with $A(3)$ obtains a solution value of zero in 0.1 sec using 18 nodes. Using $A(2)$, the solution of 1.0 is obtained in 0.1 sec and 110 nodes. For another example we use the complement of the graph G_{17} from [17]. This graph was designed with the specific intent of being difficult for methods based on semidefinite programming to obtain tight bounds on the max stable set, equivalent to the max clique for the complementary graph. For the complementary graph $\bar{G}_{17}$ the max clique has $\omega = 6$, and using $A(6)$ in MILP$_{\mathcal{COP}}$ obtains the solution value of zero in 0.16 sec using 289 nodes. Using $A(5)$, the solution value of 1.0 is obtained in 0.16 sec and 247 nodes. In the computational results of [18] none of the methods employed were able to verify copositivity of the matrix $A(6)$ associated with $\bar{G}_{17}$.

When $A(k) = k(E - \mathcal{I}_G) - E$ we know that for $k < \omega$ we have $A(k) \notin \mathcal{COP}^n$, so certainly

$A(k) \notin \mathcal{SPN}^n$. An interesting question is whether or not $A(k) \in \mathcal{SPN}^n$ for $k \geq \omega$, which can be checked by solving a conic optimization problem. Since determining the clique number is NP-complete, we would expect that in general $A(k) \notin \mathcal{SPN}^n$ for $k = \omega$. In our computations we find that this is almost always the case, and usually $A(k) \in \mathcal{SPN}^n$ for $k > \omega$. The graph $\bar{G}_{17}$ is an example where $A(k) \notin \mathcal{SPN}^n$ for $k = \omega + 1 = 7$.

We next consider the clique number for random graphs on n nodes where each edge (i, j) appears independently with probability p. Such random graphs are convenient for our purposes because it is known [1] that for fixed p, the max clique size ω will be close to the "threshold value"

$$d(n, p) = 2 \log_{1/p} n - 2 \log_{1/p} \log_{1/p} n + 2 \log_{1/p} \frac{e}{2} + 1 \tag{2}$$

with probability going to 1.0 as $n \to \infty$. For a given random graph G on n vertices, generated with edge probabilities p, we can then solve a sequence of problems $\text{MILP}_{\mathcal{COP}}$ for k in an interval around $\lfloor d(p, n) \rfloor$. For a large enough interval, with high probability these problems will include the instance with $k = \omega$ as well as instances with $k < \omega$ and $k > \omega$.

We first consider such random max-clique problems for $p = .5$ and $n = 30, 40, 50, 60, 70$. For each dimension n generate 25 random instances, and for each instance solve the $\text{MILP}_{\mathcal{COP}}$ problem using $A(k)$ for $k = \lfloor d(p, n) \rfloor \pm 3$. In Figures 1 and 2 we give the average time and nodes, respectively, to solve these instances plotted against $k - \omega$, the difference between k and the max clique size (instances where $|k - \omega| > 3$ are omitted). For the instances where $A(k) \in \mathcal{COP}^n$ the average time and nodes is somewhat higher than for the perturbed $\mathcal{SPN}^n$ instances of the same size considered in the previous section (up to a factor of about 3 for average time, and 3.6 for average nodes). It is also clear from Figures 1 and 2 that these instances become easier to solve as k decreases below ω, consistent with results in [18], but that the problem difficulty appears to be relatively unchanged as k increases above ω.

In Figures 3 and 4 we again show results for $n = 30, 40, 50, 60, 70$ but increasing the edge probability to $p = 0.7$. Note that in both figures the y axis now uses a logarithmic scale. The qualitative relationship between $k - \omega$ and difficulty is similar to that observed when $p = 0.5$, but there is a substantial increase in average time and nodes for each n compared to the case when $p = 0.5$. These increases grow with the dimension n, and for the instances with $n = 70$ and $k = \omega$ the average number of nodes increases by a factor of almost 40, and the average time increases by a factor of almost 50.

In addition to the random instances described above we considered several max-clique problems from the second DIMACS challenge. Results for these problems are given in Table 3. Several of these problems were also considered in [18]: for hamming6-4 and johnson8-2-4 both the

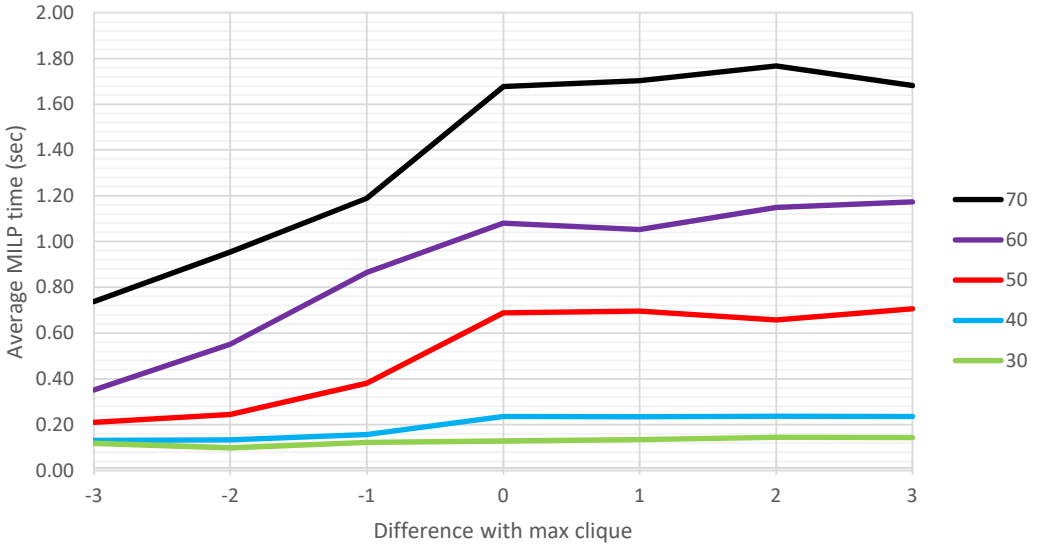

Figure 1: Average time to solve random max-clique MILP problems, $p = 0.5$

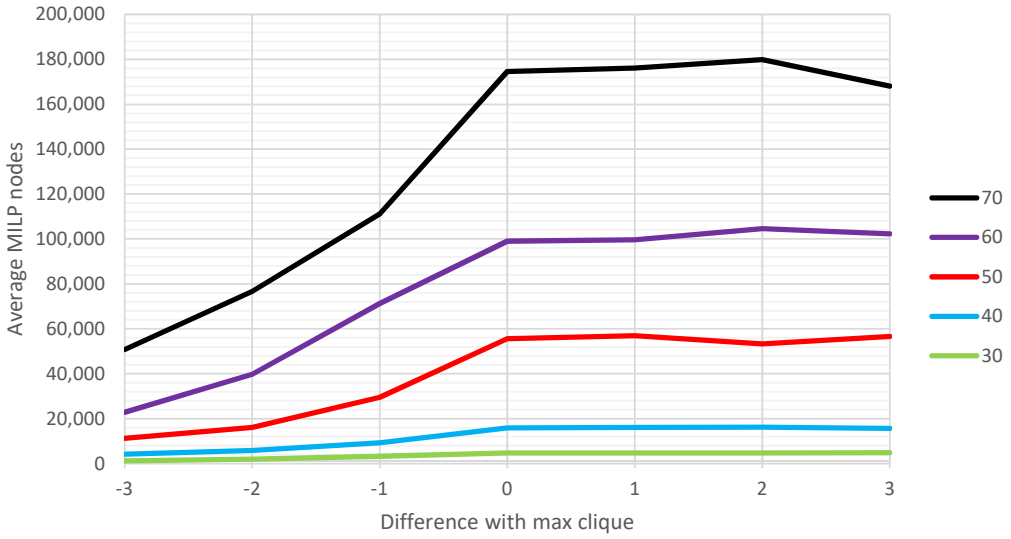

Figure 2: Average nodes to solve random max-clique MILP problems, $p = 0.5$

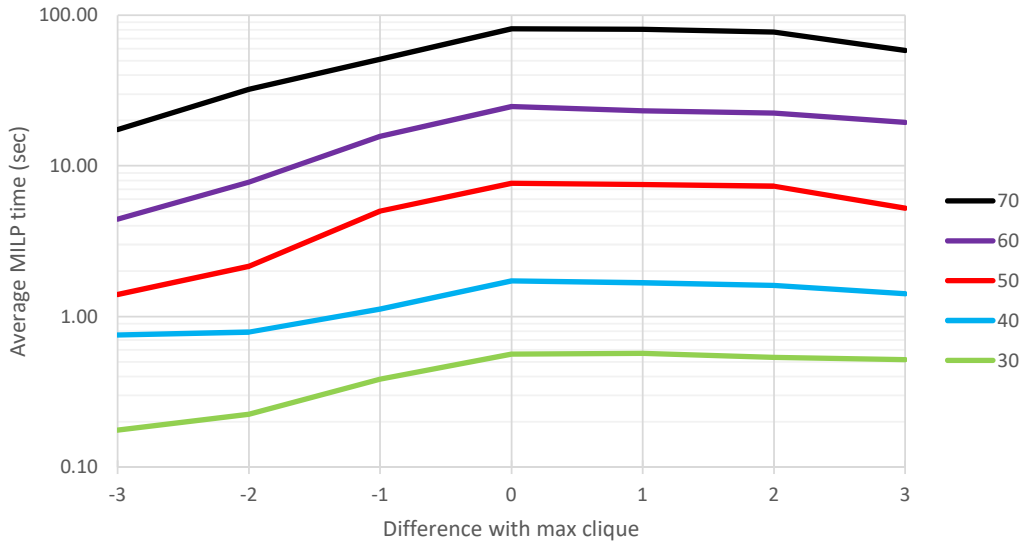

Figure 3: Average time to solve random max-clique MILP problems, $p = 0.7$

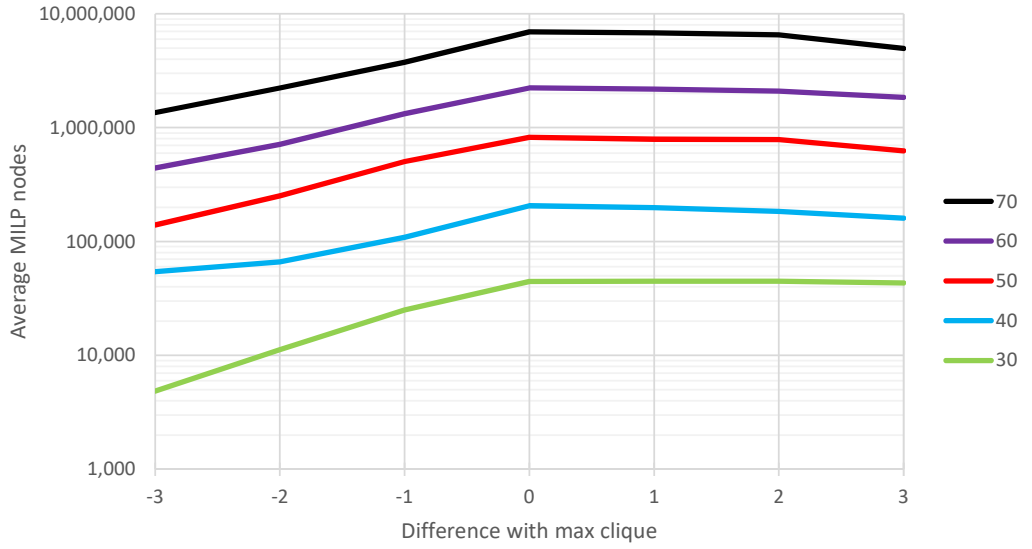

Figure 4: Average nodes to solve random max-clique MILP problems, $p = 0.7$

Table 3: Results on DIMACS max-clique problems

				$k = \omega$		$k = \omega - 1$	
Problem	n	Edges	ω	Time	Nodes	Time	Nodes
c-fat200-1	200	1534	12	0.43	9,988	0.44	7,485
c-fat200-2	200	3235	24	1.75	46,640	1.81	33,734
c-fat200-5	200	8473	58	2.02	54,951	1.17	36,573
hamming6-2*	64	1824	32	4.10	56,578	4.38	62,775
hamming6-4	64	704	4	0.13	138	0.11	132
johnson8-2-4	28	210	4	0.12	99	0.09	105
johnson8-4-4	70	1855	14	2.82	167,850	0.30	12,246
johnson16-2-4	120	5460	8	0.29	2,179	0.34	1,411
keller4	171	9435	11	48.46	2,236,156	21.16	1,259,059
MANN_a9	45	918	16	78.69	9,198,587	41.30	2,580,561

*using $\theta = k$; see text

problems with $k = \omega$ and $k = \omega - 1$ were successfully solved; for hamming6-2 and johnson8-4-4 only the problem with $k = \omega$ was sucessfully solved, and for MANN_9 only the problem with $k = \omega - 1$ was successfully solved. In the results reported in Table 3 all of the computations used MILP$_{\mathcal{COP}}$ with $\theta = 2$ except for the problem hamming6-2. We originally solved hamming6-2 for $k = \omega$ and $k = \omega - 1$ using $\theta = 2$, but these runs required a very long time and number of nodes (for $k = \omega$ over 300,000,000 nodes and almost 13 hours). Looking at the problem statistics, it is clear that the graph G for hamming6-2 is very dense (edge density over 90%). The formulation of MILP$_{\mathcal{COP}}$ with $\theta = 2$ does not seem to perform well with this level of edge density; note that the results for MANN_9 are also relatively poor given the size of the problem, and the graph for this instance is also very dense (edge density 93%).

For the problems MILP$_{\mathcal{COP}}$ arising from max-clique matrices $A(k)$ we can give a version of Theorem 2 that imposes a stronger lower bound on $e^T y$, as shown below. This modification of MILP$_{\mathcal{COP}}$ improves computational performance substantially on some instances, including hamming6-2 and MANN_9.

Theorem 3. *For a graph G on n nodes let $A(k) = k(E - \mathcal{I}_G) - E$, and let $\gamma(k)$ be the solution value in* MILP$_{\mathcal{COP}}$ *using $A = A(k)$ and $\theta = k$. Then $A(k) \in \mathcal{COP}^n$ iff $\gamma(k) = 0$.*

Proof. From Theorem 2 we know that for $k \geq \omega$ the solution value in MILP$_{\mathcal{COP}}$ using $\theta = 1$ is zero, implying that $\gamma(k) = 0$ for $k \geq \gamma$. To complete the proof we need to show that $\gamma(k) > 0$ for $k < \omega$. Let $\mathcal{C}$ be a clique in G of size ω, and let $x_i = y_i = 1$ for $i \in \mathcal{C}$, $x_i = y_i = 0$ for $i \notin \mathcal{C}$.

Then for $i \in \mathcal{C}$,

$$\sum_{j=1}^{n} A(k)_{ij} x_j = \sum_{j \in \mathcal{C}} (k(E - \mathcal{I}_G) - E)_{ij} = (k-1) - (\omega - 1) = k - \omega < 0,$$

so the constraints of MILP$_{\mathcal{COP}}$ are satisfied for $i \in \mathcal{C}$ using $\gamma = \omega - k > 0$. If necessary scaling (x, γ), we then obtain a feasible solution to MILP$_{\mathcal{COP}}$ with $\gamma > 0$. □

The results for the problem hamming6-2 in Table 3 use MILP$_{\mathcal{COP}}$ with $\theta = k$, as in Theorem 3, and this change has a dramatic effect on performance. The results for the problem MANN_a9 reported in Table 3 are based on $\theta = 2$, but we also solved instances of MILP$_{\mathcal{COP}}$ for this problem using $\theta = k$; for $k = \omega$ this reduces the time to 9.03 sec and the nodes to 274,001. It is also worth noting that to determine the max clique for a given graph G, it is actually only necessary to solve one problem of the form MILP$_{\mathcal{COP}}$. In particular, the same argument used in the proof of Theorem 3 also shows that for any $k < \omega$, for m sufficiently large the solution value in MILP$_{\mathcal{COP}}$ using $A(k)$ is exactly equal to $\omega - k$. Of course this result requires that for a given $k < \omega$, MILP$_{\mathcal{COP}}$ be solved to optimality (or at least to a tolerance sufficient to establish the optimal integer value of the solution), whereas $A(k) \notin \mathcal{COP}^n$ requires only a feasible solution of MILP$_{\mathcal{COP}}$ with $\gamma > 0$.

Acknowledgement

This paper was written while the author was visiting the Vienna Center for Operations Research at the University of Vienna, Austria. Support from VCOR is gratefully acknowledged.

References

[1] B. Bollobás and P. Erdös. Cliques in random graphs. *Math. Proc. Camb. Phil. Soc.*, 80:419–427, 1976.

[2] I. Bomze, W. Schachinger, and G. Uchida. Think co(mpletely)positive! Matrix properties, examples and a clustered bibliography on copositive optimization. *J. Global Optim.*, 52:423–445, 2012.

[3] I. M. Bomze and E. de Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *J. Global Optim.*, 24(2):163–185, 2002.

[4] S. Bundfuss and M. Dür. Algorithmic copositivity detection by simplicial partition. *Linear Algebra Appl.*, 428(7):1511–1523, 2008.

[5] S. Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Math. Program.*, 120:479–495, 2009.

[6] S. Burer and D. Vandenbussche. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Math. Program.*, 113:259–282, 2008.

[7] S. A. Burer. A gentle, geometric introduction to copositive optimization. *Math. Program. B*, 151:89–116, 2015.

[8] E. de Klerk and D. V. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM J. Optim.*, 12(4):875–892, 2002.

[9] P. J. Dickinson. A new certificate for copositivity. *Linear Algebra Appl.*, 569:15–37, 2019.

[10] P. J. C. Dickinson and L. Gijben. On the computational complexity of membership problems for the completely positive cone and its dual. *Comput. Optim. Appl.*, 57:403–415, 2014.

[11] M. Dür. Copositive Programming—A Survey. In M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels, editors, *Recent Advances in Optimization and its Applications in Engineering*, pages 3–20. Springer, 2010.

[12] K.-P. Hadeler. On copositive matrices. *Linear Algebra Appl.*, 49:79–89, 1983.

[13] J. B. Hiriart-Urruty and A. Seeger. A variational approach to copositive matrices. *SIAM Review*, 52(4):593–629, November 2010.

[14] T. Motzkin and E. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canadian J. Math.*, 17:533–540, 1965.

[15] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Math. Program.*, 39(2):117–129, 1987.

[16] J. Nie, Z. Yang, and X. Zhang. A complete semidefinite algorithm for detecting copositive matrices and tensors. *SIAM J. Optim.*, 28:2902–2921, 2018.

[17] J. Peña, J. Vera, and L. F. Zuluaga. Computing the stability number of a graph via linear and semidefinite programming. *SIAM J. Optim.*, 18:87–105, 2007.

[18] J. Sponsel, S. Bundfuss, and M. Dür. An improved algorithm to test copositivity. *J. Global Optim.*, 52:537–551, 2012.

[19] H. Väliaho. Almost copositive matrices. *Linear Algebra Appl.*, 116:121–134, 1989.

[20] D. Vandenbussche and G. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Math. Program.*, 102:559–575, 2005.