SIAM J. OPTIM.
Vol. 12, No. 4, pp. 928–948
© 2002 Society for Industrial and Applied Mathematics

A COMPLEMENTARY PIVOTING APPROACH TO THE MAXIMUM WEIGHT CLIQUE PROBLEM*

ALESSIO MASSARO†, MARCELLO PELILLO‡, AND IMMANUEL M. BOMZE§

Abstract. Given an undirected graph with positive weights on the vertices, the maximum weight clique problem (MWCP) is to find a subset of mutually adjacent vertices (i.e., a clique) having largest total weight. The problem is known to be NP-hard, even to approximate. Motivated by a recent quadratic programming formulation, which generalizes an earlier remarkable result of Motzkin and Straus, in this paper we propose a new framework for the MWCP based on the corresponding linear complementarity problem (LCP). We show that, generically, all stationary points of the MWCP quadratic program exhibit strict complementarity. Despite this regularity result, however, the LCP turns out to be inherently degenerate, and we find that Lemke's well-known pivoting method, equipped with standard degeneracy resolution strategies, yields unsatisfactory experimental results. We exploit the degeneracy inherent in the problem to develop a variant of Lemke's algorithm which incorporates a new and effective "look-ahead" pivot rule. The resulting algorithm is tested extensively on various instances of random as well as DIMACS benchmark graphs, and the results obtained show the effectiveness of our method.

Key words. maximum weight clique, linear complementarity, pivoting methods, quadratic programming, combinatorial optimization, heuristics

AMS subject classifications. 90C27, 90C20, 90C33, 90C49, 90C59, 05C69

PII. S1052623400381413

1. Introduction.

Given an undirected graph, the maximum clique problem (MCP) consists of finding a subset of pairwise adjacent vertices (i.e., a *clique*) having largest cardinality. The problem is known to be NP-hard for arbitrary graphs and, according to recent theoretical results, so is the problem of approximating it within a constant factor. An important generalization of the MCP arises when positive weights are associated to the vertices of the graph. In this case the problem is known as the maximum weight clique problem (MWCP) and consists of finding a clique in the graph which has largest total weight. (Note that the maximum weight clique does not necessarily have largest cardinality.) It is clear that the classical unweighted version is a special case in which the weights assigned to the vertices are all equal. As an obvious corollary, the MWCP has at least the same computational complexity as its unweighted counterpart. The MWCP has important applications in such fields as computer vision, pattern recognition, and robotics, where weighted graphs are employed as a convenient means of representing high-level pictorial information (see, e.g., [17, 28]). We refer to [4] for a recent review concerning algorithms, applications, and complexity issues of this important problem.

Inspired by a classical result in graph theory contributed by Motzkin and Straus [24], Gibbons et al. [13] have recently formulated the MWCP in terms of a *standard quadratic optimization problem* (StQP), which consists of minimizing a quadratic form

*Received by the editors November 17, 2000; accepted for publication (in revised form) May 21, 2001; published electronically March 13, 2002.
http://www.siam.org/journals/siopt/12-4/38141.html

†FEI Electron Optics, SEM Software Group, Building HA F13, Postbus 218, 5600MD Eindhoven, The Netherlands (alessio.massaro@nl.feico.com).

‡Dipartimento di Informatica, Università Ca' Foscari di Venezia, Via Torino 155, I-30172 Venezia Mestre, Italy (pelillo@dsi.unive.it).

§Institut für Statistik und Decision Support Systems, Universität Wien, Universitätsstraße 5, A-1010 Wien, Austria (immanuel.bomze@univie.ac.at).

over the standard simplex [3]. As shown in [7], however, their original formulation suffers from the presence of "spurious" solutions, namely, solutions of the continuous problem that are not in one-to-one correspondence with solutions in the original combinatorial problem. To avoid this drawback, in [3, 7] a new regularized quadratic programming formulation is proposed in which local and global solutions are characterized in terms of cliques of maximal and maximum weight, respectively, and no spurious solutions exist. A further benefit of this modified formulation, as we will show in this paper, is that generically all of its Karush–Kuhn–Tucker (KKT) points exhibit strict complementarity. This is a regularity property which not only favors numerical stability but also plays an important role in simplifying (second-order) optimality conditions.

It is well known that KKT points of quadratic optimization problems with linear constraints, like StQPs, can be characterized as the solutions of a linear complementarity problem (LCP), a class of inequality systems for which a rich theory and a large number of algorithms have been developed [11]. Hence, once the MWCP is formulated in terms of an StQP, the use of LCP algorithms naturally suggests itself, and this is precisely the main idea proposed in the present paper. Among the many LCP methods presented in the literature, pivoting procedures are widely used, and within this class Lemke's method is certainly the best known. Unfortunately, like other pivoting schemes, its finite convergence is guaranteed only for nondegenerate problems, and ours is indeed degenerate. To avoid this drawback, we incorporated standard degeneracy resolution strategies into Lemke's "Scheme I" procedure and tested it over a number of DIMACS benchmark graphs, but the computational results obtained were rather discouraging. The inherent degeneracy of the problem, however, is beneficial as it leaves freedom in choosing the blocking variable, and we exploit this property to develop a variant of Lemke's algorithm which uses a new and effective "look-ahead" pivot rule. The procedure depends critically on the choice of a vertex in the graph which identifies the second blocking variable in the pivoting process. Since there is no obvious way to determine such a vertex in an optimal manner, we resort to iterating this procedure over most, if not all, vertices in the graph. Also, upon analyzing the overall behavior of our heuristic, we obtain a number of invariants which are exploited to reduce the amount of data and the complexity of certain operations needed to process the problem.

The paper is organized as follows. In section 2 we review and investigate the reformulation of the MWCP as an StQP such that maximal cliques correspond to local solutions, and vice versa. Further, we establish that, for an open and dense set of weights, for a given graph all KKT points are strictly complementary. The relevance of this property becomes even more obvious in light of the discussion of second-order optimality conditions for StQPs, which we include for background information in an appendix. In the present context it is important to discriminate between strict complementarity (as a sort of "geometric" regularity condition) and the LCP degeneracy (which can be viewed as "algebraic"). The latter is shown to be inherent in the LCPs emerging from our MWCP in section 3, where we also describe our pivoting-based heuristic. Section 4 contains experimental findings. We test our approach on unweighted DIMACS benchmark graphs and various types of randomly generated weighted graphs. The results obtained show the effectiveness of our method and its clear superiority compared to other continuous-based heuristics. It also compares well with other state-of-the-art (non–continuous-based) heuristics presented in the literature.

2. Continuous formulation of the MWCP.

2.1. Basic theory. Let $G = (V, E, w)$ be an arbitrary undirected and weighted graph, where $V = \{1, \ldots, n\}$ is the vertex set and $E \subseteq \binom{V}{2}$ is the edge set, $\binom{V}{2}$ denoting the system of all two-element subsets of V. Further, $w \in \mathbb{R}^n$ is the *weight* vector, the ith component of which corresponds to the weight assigned to vertex i. It is assumed that $w_i > 0$ for all $i \in V$. Two distinct vertices $i, j \in V$ are said to be *adjacent* if they are connected by an edge, i.e., if $\{i, j\} \in E$. The *neighborhood* of a vertex i will be indicated with $N(i) = \{j \in V : \{i, j\} \in E\}$, and its degree will be $\deg(i) = |N(i)|$, the cardinality of $N(i)$. Given a subset of vertices S, the weight assigned to S will be denoted by

$$W(S) = \sum_{i \in S} w_i.$$

As usual, the sum over the empty index set is defined to be zero.

A *clique* is a subset of V in which all vertices are pairwise adjacent. A clique S is called *maximal* if no strict superset of S is a clique. A maximal weight clique S is a clique which is not contained in any other clique having weight larger than $W(S)$. Since we are assuming that all weights are positive, it is clear that the concepts of maximal and maximal weight clique coincide; hence we shall not make any distinction between these throughout the paper. A maximum cardinality clique (or, simply, a *maximum clique*) is a clique whose cardinality is the largest possible. The maximum size of a clique in G is called the *clique number* (of G) and is denoted by $\omega(G)$. A *maximum weight clique* is a clique having largest total weight, and the maximum weight clique problem (MWCP) is the problem of finding such a clique. The *weighted clique number* of G, denoted by $\omega(G, w)$, is the maximum weight of a clique in G.

Let $G = (V, E)$ be an undirected (unweighted) graph, and let Δ denote the standard simplex in the n-dimensional Euclidean space $\mathbb{R}^n$:

$$\Delta = \left\{x \in \mathbb{R}^n : x_i \geq 0 \text{ for all } i \in V,\ e^T x = 1\right\},$$

where e is a vector of appropriate length, consisting of unit entries. (Hence $e^T x = \sum_{i \in V} x_i$.) We will also denote by e_i the ith column of the $n \times n$ identity matrix I_n.

Now consider the following quadratic function, which is sometimes called the *Lagrangian* of G:

$$g(x) = x^T A_G x = \sum_{\{i,j\} \in E} x_i x_j,$$

where $A_G = (a_{ij})_{i,j \in V}$ is the adjacency matrix of G—i.e., $a_{ij} = 1$ if $\{i, j\} \in E$, and $a_{ij} = 0$ if $\{i, j\} \notin E$—and let x^* be a global maximizer of g in Δ. Motzkin and Straus [24] showed that the clique number $\omega(G)$ of G is related to $g(x^*)$ according to the following formula:

$$\omega(G) = \frac{1}{1 - g(x^*)}.$$

Additionally, they proved that a subset of vertices S is a maximum clique of G if and only if its *characteristic vector* x^S, which is the vector in Δ defined by $x_i^S = 1/|S|$ if $i \in S$ and $x_i^S = 0$ otherwise, is a global maximizer of g in Δ.[1]

[1]Actually, in their original paper, Motzkin and Straus proved just the "only if" part of this theorem. The converse direction is, however, a straightforward consequence of their result [27].

Gibbons et al. [13] have generalized the Motzkin–Straus theorem to the weighted case. Given a weighted graph $G = (V, E, w)$, they introduced the concept of the *weighted characteristic vector* $x^{S,w} \in \Delta$ for a given vertex-set $S \subseteq V$, whose coordinates are

$$x_i^{S,w} = \begin{cases} \frac{w_i}{W(S)} & \text{if } i \in S\,, \\ 0 & \text{otherwise.} \end{cases}$$

Using a proof technique suggested by Lovász, they reformulated the Motzkin–Straus problem as a minimization problem and extended the correspondence between global minimizers that have the form of weighted characteristic vectors and the maximum weight cliques of G. Their results were proved over a whole class of matrices, rather than just a single matrix as in the original Motzkin–Straus formulation. Of course, both the latter and the matrix class considered in [13] depend on G.

However, the formulation of Gibbons et al. has a major drawback which, as in the unweighted case [27], relates to the presence of "spurious" solutions, i.e., local or global solutions that are not in the form of weighted characteristic vectors $x^{S,w}$ for some subset S of vertices. (See [7] for an in-depth study on this topic.) Even though in certain specific circumstances such solutions may provide useful information concerning the structure of the underlying graph, computationally they represent a nuisance, for we cannot extract the vertices comprising the clique directly from them; they just provide information about the weighted clique number.

This problem is solved in [3] by considering the matrix $Q_G = [q_{ij}]_{i,j \in V \times V}$ defined as

$$q_{ij} = \begin{cases} \frac{1}{2w_i} & \text{if } i = j, \\ 0 & \text{if } \{i,j\} \in E, \\ \frac{1}{2w_i} + \frac{1}{2w_j} & \text{otherwise} \end{cases} \tag{1}$$

and investigating the StQP

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Delta \end{array} \tag{2}$$

with

$$f(x) = x^T Q_G x. \tag{3}$$

Indeed, a whole class of matrices serves the same purpose again; this class, of course, differs from that used in [13].

The following theorem proved in [3] summarizes the result around which our work is centered.

THEOREM 2.1. *Let $G = (V, E, w)$ be an arbitrary graph with positive weight vector $w \in \mathbb{R}^n$, and consider problem* (2). *Then the following assertions hold.*

- *A vector $x \in \Delta$ is a local solution of* (2) *if and only if $x = x^{S,w}$, where S is a maximal clique of G.*
- *A vector $x \in \Delta$ is a global solution of* (2) *if and only if $x = x^{S,w}$, where S is a maximum weight clique of G.*

Moreover, all solutions of (2) *are strict.*

2.2. From KKT points to maximal cliques. It is a simple exercise to show that the KKT first-order optimality conditions for a point $x \in \Delta$ in program (2), with a general symmetric $n \times n$ matrix Q in place of Q_G, can be written as

$$(Qx)_i \begin{cases} = \lambda & \text{if } x_i > 0, \\ \geq \lambda & \text{if } x_i = 0, \end{cases} \tag{4}$$

for some real-valued constant λ.

Of course, a KKT point is not necessarily a local solution of (2), and hence a maximal clique of G, but in light of (4) it is possible to derive some useful properties that virtually eliminate the need to guarantee local optimality and give direct methods to attain it once a KKT point is available. For $x \in \Delta$, let us denote by

$$S(x) = \{i \in V : x_i > 0\}$$

the support of x.

THEOREM 2.2. *Let $G = (V, E, w)$ be a weighted graph and x a KKT point of* (2). *If $C = S(x)$ is a clique of G, then C is a maximal clique.*

Proof. If $C = S(x)$ is a nonmaximal clique, then there exists a $k \notin C$ such that $(i, k) \in E$ for all $i \in C$. For such a k we have

$$(Q_G x)_k = \sum_{j \in C} q_{kj} x_j = 0.$$

On the other hand, for any $i \in C$, we have

$$(Q_G x)_i = \sum_{j \in C} q_{ij} x_j = q_{ii} x_i > 0,$$

which contradicts the hypothesis that x is a KKT point for (2). □

The practical significance of Theorem 2.2 reveals itself in large graphs: Even if these are quite dense, cliques are usually much smaller than the graph itself. Now suppose we are returned a KKT point x by some method. Then we set $C = S(x)$ and check whether or not C is a clique. This requires $\mathcal{O}(s^2)$ steps if C contains s vertices, while checking whether this clique is maximal would require $\mathcal{O}(sn)$ steps and, as stressed above, usually $s \ll n$. But Theorem 2.2 now guarantees that the obtained clique C (if it is one) must automatically be maximal, and thus we are spared from trying to add external vertices. But how should one behave in the case of a nonclique KKT point? The answer is to be found in part of the proof of Theorem 5 in [13] and is summarized in the following result.

THEOREM 2.3. *Let $G = (V, E, w)$ be a weighted graph and x a KKT point of* (2) *with support $C = S(x)$. If $i, j \in V$ are two nonadjacent vertices of C, then $x_\delta = x + \delta\,(e_i - e_j)$ improves the objective function f of* (2)*; i.e., $f(x_\delta) < f(x)$ for any $0 < \delta \leq x_j$.*

Proof. From the symmetry of Q_G, we have

$$\begin{aligned} f(x_\delta) &= (x + \delta\,(e_i - e_j))^T Q_G\,(x + \delta\,(e_i - e_j)) \\ &= x^T Q_G x + 2\delta\,(e_i - e_j)^T Q_G x + \delta^2\,(e_i - e_j)^T Q_G\,(e_i - e_j)\,. \end{aligned}$$

Since x is a KKT point, the second term is null, and hence

$$f(x_\delta) = f(x) + \delta^2\,(q_{ii} + q_{jj} - 2q_{ij})\,.$$

But $\{i,j\} \notin E$ implies that $q_{ii} + q_{jj} - 2q_{ij} < 0$, and this concludes the proof. ∎

Once a KKT point has been obtained by some method, the most effective way to use Theorem 2.3 is to check for pairs $\{i,j\} \notin E$ with $\{i,j\} \subseteq C$. If there are none, C is a clique and hence a maximal clique. Otherwise, choose any pair of nonadjacent vertices in C and construct a new "better" point as described in Theorem 2.3. The proof of the theorem also provides us with a criterion to determine the best such points, namely, the one which minimizes $x_j^2\,(q_{ii} + q_{jj} - 2q_{ij})$. This can be done very quickly in $\mathcal{O}\left(\binom{m}{2}\right)$ time, where $m = |C|$.

Clearly, the new improved point does not necessarily correspond to a (maximal) clique, but by iterating this procedure, as suggested in [20] for the unweighted case, we can readily obtain one. Alternatively, and more interestingly, one can give the new improved point as input to any gradient-based technique. These are typically very efficient in terms of computation time and can be quite effective if kick-started from within a close range to a good suboptimal solution. An example of such techniques is given by the so-called replicator dynamics, a class of dynamical systems developed and studied in evolutionary game theory [15]. We refer to [26] for a recent review concerning the application of these dynamics to combinatorial optimization, and to [22, 23] for independent connections between this kind of dynamical equations and LCPs.

2.3. Strict complementarity is generic. We close this section by establishing easy-to-check regularity conditions for the StQP (2) based on the matrix Q_G, which ensure the strict complementarity of all KKT points of this StQP. It turns out that, when we fix the discrete structure (V,E) of the graph in an arbitrary way, strict complementarity holds for a set of weights w that is an open and dense subset of the positive orthant $\mathbb{R}^n_+$. Recall that a KKT point x satisfies the *strict complementarity condition* for the StQP (2), with a general symmetric $n \times n$ matrix Q in place of Q_G, if and only if all Lagrange multipliers are strictly positive: $\lambda_i > 0$ for all $i \in V \setminus S(x)$, where $\lambda_i = (Qx)_i - \lambda$ from (4).

We now characterize strict complementarity and establish easy-to-check sufficient conditions.

THEOREM 2.4. *Let $x \in \Delta$ be a KKT point for (2), and again set $S(x) = \{i \in V : x_i > 0\}$ as well as $T(x) = \{i \in V : (Qx)_i = x^TQx\}$. Then $S(x) \subseteq T(x)$. Further, the following assertions are equivalent:*

(a) *$S(x) = T(x)$ (which in particular holds true if $S(x) = V$);*

(b) *x satisfies the strict complementarity condition.*

Both conditions are met if for all $i \in V \setminus S(x)$ the matrices

$$Q_{S(x)}(i) = [q_{kj} - q_{ij}]_{(k,j) \in S(x) \times S(x)}$$

are nonsingular.

Proof. The inclusion $S(x) \subseteq T(x)$ is nothing other than (4); indeed, it easily follows that $\lambda = x^TQx$. Further, we also get $0 \leq \lambda_i = (Qx)_i - \lambda = (Qx)_i - x^TQx$, from which the equivalence of (a) and (b) is immediate. Finally, suppose that there is an index $i \in T(x) \setminus S(x) \subseteq V \setminus S(x)$. Then we get $Q_{S(x)}(i)x_{S(x)} = [(Qx)_k - (Qx)_i]_{k \in S(x)} = o$ while $x_{S(x)} = [x_k]_{k \in S(x)} \neq o$, contradicting the assumption. ∎

Now we are ready to establish the main result for matrix $Q = Q_G$ used in the MWCP treatment; for almost all weights w in a given graph G, every KKT point has this property. We also specify simple explicit sufficient conditions which guarantee this.

THEOREM 2.5. *Let $G = (V, E, w)$ be a weighted graph and suppose that $w = \mu z$, where $\mu > 0$ and $z_i > 0$ are odd integers for all $i \in V$. Then, regardless of the structure of G, all the matrices $Q_{S(x)}(i)$ originating from the matrix $Q = Q_G$ are nonsingular, and hence all KKT points x for (2) satisfy strict complementarity.*

Further, the set of all weights such that the nonsingularity condition (and thus strict complementarity) holds is open and dense in $\mathbb{R}^n_+$.

Proof. As is easily seen, the entries of $Q_{S(x)}(i)$ all are sums of two terms belonging to the set $\{0, \pm\frac{1}{2w_j} : j \in V\}$. Hence multiplication of the weights w_j by a common factor μ does not alter any aspect of the assertion. Thus the result holds for all μ, given that we establish it for a special value of it, e.g., for $\mu = [2\prod_{i\in V} z_i]^{-1}$. But then $Q_{S(x)}(i)$ has odd integer diagonal entries while all other entries are even integers. Thus the determinant is an odd number, whence it follows that $Q_{S(x)}(i)$ is nonsingular. Turning to the genericness assertion, openness is clear from the continuity of the determinant, while denseness follows from an approximation argument; indeed, every positive w can be arbitrarily well approximated by a vector with positive rational entries n_i/d, where n_i and d are positive integers. Next choose an integer K large enough such that these ratios n_i/d in turn are close to $\tilde{w}_i = \mu(2Kn_i + 1)$ with $\mu = [2Kd]^{-1}$. Now $\tilde{w}$ satisfies the first condition of the theorem, and the result follows. ▯

Note that the result applies particularly to the nonweighted case; in fact, $w = e$ satisfies the first condition in the above theorem.

In spite of these results, namely, that this "geometric" form of degeneracy is highly unlikely, we will see in the next section that a sort of "algebraic" degeneracy is inherent to the problem class considered here. To promote the flow of the argument, we defer to an appendix a discussion of further aspects of strict complementarity in relation to the optimality condition.

3. Complementary pivoting.

3.1. Lemke's method. The KKT points of (2) can be computed by solving the LCP (q_G, M_G), which is the problem of finding a vector x satisfying the system

$$y = q_G + M_G x \geq 0, \quad x \geq 0, \quad x^T y = 0, \tag{5}$$

where

$$q_G = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 1 \end{bmatrix}, \quad M_G = \begin{bmatrix} Q_G & -e & e \\ e^T & 0 & 0 \\ -e^T & 0 & 0 \end{bmatrix}, \tag{6}$$

and Q_G is as in (1). With the above definitions, it is well known that if z is a complementary solution of (q_G, M_G) with $z^T = [x^T, y^T]$ and $x \in \mathbb{R}^n$, then x is a KKT point of (2). Note that Q_G is strictly $\mathbb{R}^n_+$-copositive; hence so is M_G, and this is sufficient to ensure that (q_G, M_G) always has a solution—see the fundamental book [11], where a large number of LCP algorithms can also be found. The most popular among them is probably Lemke's method, largely for its ability to provide a solution for several matrix classes. Lemke's "Scheme I" belongs to the family of pivoting algorithms. Given the generic LCP (q, M), it deals with the augmented problem

(q, d, M) defined by

$$y = q + [M, d] \begin{bmatrix} x \\ \theta \end{bmatrix} \geq 0, \quad \theta \geq 0, \quad x \geq 0, \quad x^T y = 0. \tag{7}$$

Vector d is called the *covering vector* and must satisfy $d_i > 0$ whenever $q_i < 0$. A solution of (q, d, M) with $\theta = 0$ promptly yields a solution for (q, M), and Lemke's method intends to compute precisely such a solution. We refer to [11] for a detailed description of Lemke's algorithm. In our implementation we chose $d = e$, as our problem does not expose peculiarities that would justify a deviation from this common practice.

Assuming the nondegeneracy of the LCP is a strategy commonly taken to prove the finiteness of pivoting schemes. In particular, Lemke's method is guaranteed to process any nondegenerate problem (q, M), where M is strictly $\mathbb{R}^n_+$-copositive, and to do so without terminating on a secondary ray [11].

Unfortunately (q_G, M_G) is degenerate, but it is possible to give an equivalent formulation of (2) in order to obtain a nondegenerate LCP. To this end, it is easy to see that program (2) is equivalent to the following program:

$$\begin{array}{ll} \text{minimize} & x^T \widehat{Q}_G x + c^T x \\ \text{subject to} & x \in \Delta, \end{array}$$

where $c \in \mathbb{R}^n$ and $\widehat{Q}_G = Q_G - \left(ce^T + ec^T\right)$. If $c \leq 0$, then copositivity is maintained, and if all its entries are different, then the corresponding LCP is nondegenerate. Furthermore, if $c_i \leq -1$ for some i, it is straightforward to check that even the first pivot step of Lemke's method changes.

The above method for degeneracy removal has a characteristic in common with the lexicographic degeneracy resolution method (LDR) [11]. Namely, they both require the introduction of extra data: vector c in the previous case, and a nonsingular square matrix as big as M with lexicographically positive rows in the case of LDR. Of course, these objects have to be assigned values and there are myriads of sensible methods for doing so, each one having a different theoretical ground and/or performance impact on the final result.

We report in Tables 1 and 2 (column *LDR*) the results obtained on the DIMACS benchmark graphs (see subsection 4.1) by running Lemke's "Scheme I" with LDR, using the identity matrix as extra data. Their order, density, and clique number are shown in columns *Order*, *Density*, and ω, respectively. A complete description of the table can be found in subsection 4.1. It is clear to see that in all but the most trivial cases LDR performs poorly, although it is extremely fast. The tendency of the previously discussed degeneracy treatment methods is that they lead to inefficient local minimizers, i.e., to maximal cliques of small size.

3.2. A pivoting-based heuristic. Rather than continuing to investigate the enormous variety of assignment techniques for removing degeneracy mentioned in the previous subsection, we focused on examining the original degenerate form of the LCP (q_G, M_G). Such degeneracy even turns out to be beneficial for performance, since it permits freedom in choosing the blocking variable within a successful variant of Lemke's method. This is opposed to the nondegenerate version of the latter method, in which those variables are uniquely determined. This is the topic of the present subsection.

As is customary, we will use an exponent for the problem data and, to make notations simpler, we will omit subscripts indicating the dependence on graph G. Hence, q^ν and M^ν will identify the situation after ν pivots, and Q^ν will indicate the $n \times n$ leading principal submatrix of M^ν. Consistently, y^ν and x^ν will indicate the vectors of basic and nonbasic variables, respectively, each made up of a combination of the original x_i and y_i variables. The notation $\langle x_i^\nu, y_j^\nu \rangle$ will be used to indicate pivoting transformations. The index set of the basic variables that satisfy the min-ratio test at iteration ν will be denoted with Ω^ν, i.e.,

$$\Omega^\nu = \arg\min_i \left\{ \frac{-q_i^\nu}{m_{is}^\nu} : m_{is}^\nu < 0 \right\},$$

where s is the index of the driving column. Also, in what follows, the auxiliary column that contains the covering vector d in (7) will be referred to as the column $n+3$ of matrix $M = M_G$. The nondegeneracy assumption basically amounts to having $|\Omega^\nu| = 1$ for all ν, thereby excluding any cycling behavior.

Here we employ the least-index rule, which amounts to blocking the driving variable with a basic one that has minimum index within a certain subset of Ω^ν, i.e., $r = \min \Phi^\nu$ for some $\Phi^\nu \subseteq \Omega^\nu$. The set Φ^ν is chosen in order to make the number of degenerate variables decrease as slowly as possible, i.e., among the index-set

$$\Phi^\nu = \arg\min_i \left\{ |\Omega^\nu| - \left|\Omega_i^{\nu+1}\right| > 0 : i \in \Omega^\nu \right\} \subseteq \Omega^\nu,$$

where $\Omega_i^{\nu+1}$ is the index-set of those variables that would satisfy the min-ratio test at iteration $\nu + 1$ if the driving variable at iteration ν were blocked with y_i^ν as $i \in \Omega^\nu$. The previous conditional implies that a pivot step is taken and then reset in a sort of "look-ahead" fashion; hence we will refer to this rule as the *look-ahead (pivot) rule.*

Before actually proceeding to illustrate a variant of Lemke's algorithm applied to the MWCP, let us take a look at the tableaus that it generates. This will help us to identify regularities that are reflected in the behavior of the algorithm itself. The initial tableau follows:

(8)

	q	x_1	$\cdots$	x_n	x_{n+1}	x_{n+2}	θ
y_1	0				-1	1	1
$\vdots$	$\vdots$		Q_G		$\vdots$	$\vdots$	$\vdots$
y_n	0				-1	1	1
y_{n+1}	-1	1	$\cdots$	1	0	0	1
y_{n+2}	1	-1	$\cdots$	-1	0	0	1

.

As q_{n+1} is the only negative entry for the column of q, the first pivot to occur during initialization is $\langle y_{n+1}, \theta \rangle$, thereby producing the following transformation:

(9)

	q	x_1	$\cdots$	x_n	x_{n+1}	x_{n+2}	y_{n+1}
y_1	1				-1	1	1
$\vdots$	$\vdots$		$Q_G - ee^T$		$\vdots$	$\vdots$	$\vdots$
y_n	1				-1	1	1
θ	1	-1	$\cdots$	-1	0	0	1
y_{n+2}	2	-2	$\cdots$	-2	0	0	1

.

The driving variable for the second pivot is x_{n+1}. Since $m_{i,n+1}^1 = -1$ for all $i = 1, \ldots, n$, it is clear to see that the relative blocking variable can be any one of

Algorithm 3.1. Lemke's "Scheme I" with the "look-ahead" rule applied to the MWCP.

```
Input: A graph $G = (V, E, w)$ and $p \in V$.
Let $(q_G, e, M_G)$ be the augmented LCP, where $q_G$ and $M_G$ are defined in (6).
$\nu \leftarrow 0$, perform $\langle y_{n+1}, \theta\rangle$ and $\langle y_p, x_{n+1}\rangle$.
The driving variable is $x_p$.
Infinite loop
        $\nu \leftarrow \nu + 1$.
        Let $x_s^\nu$ denote the driving variable.
        $\Omega^\nu = \arg\min_i\{-q_i^\nu/m_{is}^\nu \ : m_{is}^\nu < 0\}$.
        If $|\Omega^\nu| = 1$, then $r = \min \Omega^\nu$;
        else $\Phi^\nu = \arg\min_i\{|\Omega^\nu| - |\Omega_i^{\nu+1}| > 0 : i \in \Omega^\nu\}$, $r = \min \Phi^\nu$.
        Perform $\langle y_r^\nu, x_s^\nu\rangle$.
        If $y_r^\nu \equiv \theta$, then:
            Let $x$ denote the complementary solution of $(q_G, M_G)$ found.
            The result is $\operatorname{supp}(x) \cap V$.
        The new driving variable is the variable complementary to $y_r^\nu$.
```

$y_1, \ldots, y_n$. In this case we apply no degeneracy resolution criterion but rather allow for user intervention by catering for the possibility of deciding the second blocking variable a priori. Thus let y_p be the (arbitrary) variable that shall block x_{n+1}. After performing $\langle y_p, x_{n+1}\rangle$, we have the following tableau:

$$\begin{array}{c|c|ccc|ccc|} & q & x_1 & \cdots & x_n & y_p & x_{n+2} & y_{n+1} \\ \hline y_1 & 0 & & & & 1 & 0 & 0 \\ \vdots & \vdots & & & & \vdots & \vdots & \vdots \\ y_{p-1} & 0 & & & & 1 & 0 & 0 \\ x_{n+1} & 1 & & Q_p & & -1 & 1 & 1 \\ y_{p+1} & 0 & & & & 1 & 0 & 0 \\ \vdots & \vdots & & & & \vdots & \vdots & \vdots \\ y_n & 0 & & & & 1 & 0 & 0 \\ \hline \theta & 1 & -1 & \cdots & -1 & 0 & 0 & 1 \\ y_{n+2} & 2 & -2 & \cdots & -2 & 0 & 0 & 1 \\ \hline \end{array}, \tag{10}$$

where Q_p denotes the matrix whose rows are defined as

$$(Q_p)_i = \begin{cases} (Q_G)_p - e^T & \text{if } i = p, \\ (Q_G)_i - (Q_G)_p, & \text{otherwise.} \end{cases}$$

Algorithm 3.1 formalizes the above statements. We now introduce a number of invariants aimed at reducing the size of the data required by the process and the complexity of certain operations.

Proposition 3.1. *Within Algorithm* 3.1*, after the first pivot and as long as none occurs within the last* 2 *rows of* M^ν*, the ratios* $m_{n+1,j}^\nu/m_{n+2,j}^\nu = q_{n+1}^\nu/q_{n+2}^\nu = \frac{1}{2}$ *for* $j = 1, \ldots, n$ *do not change.*

Proof. The proof is elementary by the definition of pivot operation and the structure of tableau (9). ∎

Corollary 3.2. *In Algorithm* 3.1*, in the event that after* ν *pivot operations the* *whole driving column of Q^ν be nonnegative, the schema will pivot on the row of θ and terminate.*

Proof. The proof follows immediately from the fact that termination on the secondary ray cannot occur, and from Proposition 3.1. ∎

PROPOSITION 3.3. *After* 2 *pivot operations within Algorithm* 3.1*, the columns of q, x_{n+2}, y_{n+1} do not change as long as no pivot is performed on the rows of x_{n+1}, θ, or y_{n+2}.*

Proof. The proof follows from tableau (10), observing that the hypothesis implies that either m^ν_{is}, m^ν_{rj}, or q^ν_r is null when calculating the successive transforms of these columns. ∎

COROLLARY 3.4. *After* 2 *pivot operations within Algorithm* 3.1*, if a pivot on the row of $x_{n+1} \equiv y^\nu_r$ occurs with x^ν_s as the driving variable, then $m^\nu_{i,s} \geq 0$ for all $i = 1, \ldots, r-1, r+1, \ldots, n$. Moreover, if $m^\nu_{n+1,s} < 0$, then $m^\nu_{n+1,s} \geq m^\nu_{r,s}$.*

Proof. If there were other negative entries for $i = 1, \ldots, r-1, r+1, \ldots, n$ for Proposition 3.3, they would have a null ratio. On the other hand, the ratio for the row of x_{n+1} is certainly positive. A similar argument proves the remaining part of the corollary. ∎

PROPOSITION 3.5. *After* 2 *pivot operations within Algorithm* 3.1*, pivoting on the row of x_{n+1} ends the schema with the pivot sequence $\langle x_{n+1}, x^\nu_s\rangle$, $\langle y_{n+2}, y_{n+1}\rangle$, $\langle \theta, x_{n+2}\rangle$.*

Proof. After $\langle x_{n+1}, x^\nu_s\rangle$, for Proposition 3.1 and Corollary 3.4 we have $m^\nu_{i,n+2} = m^\nu_{i,n+3} \geq 0$ for all $i = 1, \ldots, n$. Corollary 3.4 yields $m^\nu_{n+1,n+3} = 1 - m^\nu_{n+1,s}/m^\nu_{r,s} \geq 0$, and this, together with the fact that no secondary ray termination can occur, implies $m^\nu_{n+2,n+3} < 0$, thereby indicating $\langle y_{n+2}, y_{n+1}\rangle$ as the following pivot. Similar arguments prove the remaining part of the proposition. ∎

The above statements show that the $x_1, \ldots, x_n$ variables remain within the Q^ν block for the whole duration of Algorithm 3.1. Furthermore, we do not need to perform the terminal pivot sequence of Proposition 3.5 for, as soon as x_{n+1} blocks the driving variable, we know which of the x_i with $i \in V$ will be basic, and that is enough to compute the final clique. This is sufficient to derive that the rows and columns associated with the simplex constraints and the covering vector are not needed to process (q_G, M_G). On top of that, for Proposition 3.3 we can also discard the vector q and reduce the min-ratio test to a mere negativity test. All these concepts are formalized in Algorithm 3.2.

Empirical evidence indicated p as a key parameter for the quality of the final result of Algorithm 3.2. Unfortunately we could not identify any effective means to restrict the choice of values in V that can guarantee a good suboptimal solution. We thus had to consider iterating for most, if not all, vertices of V as outlined in Algorithm 3.3. Here we employ a very simple criterion to avoid considering those nodes that cannot drive to larger cliques than the one we already have, because their weights and those of their neighborhoods are too small. It is easy to comment that such a criterion is effective only for very sparse graphs.

We also observed that the schema is sensitive to the ordering of nodes and found that the best figures were obtained by reordering G by the decreasing weight of each node and its neighborhood. This feature too is formalized in Algorithm 3.3. We will refer to this scheme by the name *pivoting-based heuristic* (PBH).

Before concluding this section, it is worth mentioning the fact that we were not able to prove that Algorithms 3.1 and 3.2 cannot terminate prematurely with an empty Φ^ν set, or to loop indefinitely with Ω^ν being a singleton. However, neither

Algorithm 3.2. A reduced version of Algorithm 3.1.

```
Input: A graph $G = (V, E, w)$ and $p \in V$.
Let $Q_p = (q_{ij})$. $\nu \leftarrow 2$. $K \leftarrow \emptyset$.
The driving variable is $x_p$.
Infinite loop
        Let $x_s^\nu$ denote the driving variable.
        $\Omega^\nu = \{i : q_{is}^\nu < 0\}$.
        If $\Omega^\nu \subseteq \{p\}$, stop: the result is $K$.
        $\Phi^\nu = \arg\min_i \left\{|\Omega^\nu| - \left|\Omega_i^{\nu+1}\right| > 0 : i \in \Omega^\nu\right\}$.
        $r = \min \Phi^\nu$.
        If $y_r^\nu \equiv x_i$ for some $i$, then $K \leftarrow K \setminus \{i\}$.
        Perform $\langle y_r^\nu, x_s^\nu \rangle$.
        The new driving variable is the variable complementary to $y_r^\nu$.
        $\nu \leftarrow \nu + 1$.
        If $y_r^\nu \equiv x_i$ for some $i$, then $K \leftarrow K \cup \{i\}$.
```

Algorithm 3.3. The pivoting-based heuristic (PBH) for the MWCP.

```
Input: A graph $G = (V, E, w)$.
Let $G' = (V', E', w')$ be a permutation of $G$
with $W(u' \cup N(u')) \geq W(v' \cup N(v'))$ for all $u', v' \in V'$ with $u' < v'$.
$K^\star \leftarrow \emptyset$.
For $v' = 1, \ldots, n : W(v' \cup N(v')) > W(K^\star)$ do:
    Run Algorithm 3.2 with $G'$ and $v'$ as input.
    Let $K$ be the obtained result.
    If $W(K) > W(K^\star)$, then $K^\star \leftarrow K$.
The result is the mapping of $K^\star$ in $G$.
```

of these circumstances ever actually occurred in practice. Instead, for all the several thousand graphs we tested them on, we observed that once an x_i variable with $i \in V$ had entered the basis, it never exited it. In fact, if Algorithm 3.1 found a clique with s nodes, it always performed exactly $s + 3$ pivot steps. This fact led us to consider a simplified implementation of Algorithm 3.2 which was in fact used to produce the results presented in the following section. This simplified version simply lacks the tests to remove an x variable from the basis. A thorough empirical analysis has confirmed that both the original and simplified versions of the algorithm behave identically.

Computing $|\Omega_i^{\nu+1}|$ can be done with $\mathcal{O}(n)$ time complexity, as only the driving column is needed for this purpose, and a pivotal transformation takes $\mathcal{O}(n^2)$ computations. This, together with our previous observation, gives us strong empirical evidence that PBH is $\mathcal{O}(sn^3)$, where s is the size of the clique found. Note, however, that it is quite straightforward to parallelize the algorithm over n processors, thereby reducing its time complexity to $\mathcal{O}(sn^2)$. With respect to space complexity, our implementation was $\mathcal{O}(n^2)$, as we could not find better techniques than implementing tableau-style pivoting.

4. Experimental results.

To practically assess the effectiveness of the proposed approach, we conducted a large number of experiments. First, we focused on unweighted DIMACS graphs, which constitute a standard benchmark for clique-finding heuristics [19].[2] Next, we considered various types of randomly generated weighted graphs.

4.1. Unweighted DIMACS graphs. Tables 1 and 2 show the performance figures obtained by running PBH (column *PBH*) over a selection of DIMACS benchmark graphs. Their order, density, and clique number are reported in columns *Order*, *Density*, and ω, respectively. The column marked with *LDR* lists the results pertaining to Lemke's method with the lexicographic degeneracy resolution criterion; see subsection 3.1. Computing time (column *Time*) for LDR as well as PBH is in seconds and refers to a C++ implementation for a Linux machine with a 655MHz Celeron CPU (77MHz FSB $\times$ 8.5). Some figures are missing because the Unix `clock` system call could not time periods longer than approximately 30 minutes on our test machine.

We compare our methods with three other heuristics based, as ours is, on the Motzkin–Straus formulation. The first method considered is the *continuous-based heuristic* (CBH) of Gibbons, Hearn, and Pardalos [12], which employs a parameterized version of the original Motzkin–Straus program. The problem is divided into a series of subproblems with the simplex constraints relaxed into spherical ones. Their schema uses a combinatorial postprocessing phase to round the solutions produced by a relaxation procedure that solves the subproblems.

The second algorithm is *annealed replication* (AR) [5]. It uses a different parameterized and unweighted maximization form of problem (2) that has $x^T (A_G + \alpha I) x$ as objective function. The heuristic uses the replicator dynamics as a local search technique and is based on a proper variation of α after a model similar to simulated annealing, but it is motivated by more principled arguments.

The third method is the *RD-algorithm* (RD), a recent heuristic of Kuznetsova and Strekalovsky [20]. They approach the approximate solution of the regularized Motzkin–Straus (unweighted) program by splitting its objective function into two convex terms, for which they obtain a set of global optimality conditions. At each iteration their method improves upon a KKT point which is sought by some conventional procedure.

Before commenting on the results presented in Tables 1 and 2, we note that LDR performed very poorly in all but the most trivial instances, although it converged very quickly. In fact, it is even worse than plain replicator dynamics, which are essentially gradient-based procedures (see [6]).

The *c-fat*, *Hamming*, and *Johnson* graph categories are certainly those that have proven most vulnerable to the different approaches. All methods, in fact, managed to systematically attain a maximum clique, except for one Hamming graph. *Hamming* and *Johnson* graphs are borrowed from coding theory, whereas the *c-fat* ones are used in fault diagnosis. The notation "-" in columns AR, CBH, and RD indicates data not presented in the original papers, from which the values here were taken.

The *p_hat* graphs are generalized random graphs with a wider node degree spread. The generation procedure is described in [30]. In 10 out of 15 cases PBH produced the best results, and 6 of them were maximum cliques. The largest known clique was actually reached in 9 cases.

Graphs prefixed with *MANN* are a reduction to the MCP of the minimum set covering problem. For the two smallest problems, RD and our method performed equally well, obtaining a maximum clique and a largest maximal one. For the third, bigger problem, a clique very close to the maximum one (342 vs. 345) was obtained

[2]Data can be found at http://dimacs.rutgers.edu.

TABLE 1
Performance of LDR, PBH, and other competing heuristics on unweighted DIMACS graphs (part I*). Entries that correspond to the best result for a given graph are boldfaced.*

Graph	Ord.	Dens.	ω	AR	CBH	RD	LDR	Time	PBH	Time
c-fat200-1	200	7.7%	12	-	**12**	**12**	**12**	0.07	**12**	5.0
c-fat200-2	200	16.3%	24	-	**24**	**24**	**24**	0.12	**24**	9.0
c-fat200-5	200	42.6%	58	-	**58**	**58**	**58**	0.28	**58**	22.5
c-fat500-1	500	3.6%	14	-	**14**	**14**	**14**	0.48	**14**	100.3
c-fat500-2	500	7.3%	26	-	**26**	**26**	**26**	0.79	**26**	185.2
c-fat500-5	500	18.6%	64	-	**64**	**64**	**64**	1.83	**64**	464.5
c-fat500-10	500	37.4%	126	-	**126**	**126**	**126**	3.59	**126**	1024.2
hamming6-2	64	90.5%	32	-	**32**	**32**	**32**	0.01	**32**	0.4
hamming6-4	64	34.9%	4	-	**4**	**4**	**4**	0.00	**4**	0.1
hamming8-2	256	96.9%	128	-	**128**	**128**	**128**	0.98	**128**	252.6
hamming8-4	256	63.9%	16	-	**16**	**16**	**16**	0.14	**16**	22.8
hamming10-2	1024	99.0%	512	-	**512**	-	**512**	61.01	**512**	-
hamming10-4	1024	82.9%	$\geq$ 40	-	**35**	-	32	4.1	32	-
johnson8-2-4	28	55.6%	4	-	**4**	**4**	**4**	0.00	**4**	0.0
johnson8-4-4	70	76.8%	14	-	**14**	**14**	**14**	0.01	**14**	0.3
johnson16-2-4	120	76.5%	8	-	**8**	**8**	**8**	0.01	**8**	1.1
johnson32-2-4	496	87.9%	$\geq$ 16	-	**16**	**16**	**16**	0.54	**16**	184.8
p_hat300-1	300	24.4%	8	**8**	**8**	**8**	6	0.10	**8**	14.0
p_hat300-2	300	48.9%	25	**25**	**25**	**25**	16	0.20	**25**	34.9
p_hat300-3	300	74.5%	36	35	**36**	34	21	0.25	35	61.0
p_hat500-1	500	25.3%	9	**9**	**9**	**9**	6	0.27	**9**	83.5
p_hat500-2	500	50.5%	36	**36**	35	35	26	0.82	**36**	282.5
p_hat500-3	500	75.2%	$\geq$ 50	47	**49**	**49**	30	0.94	48	485.7
p_hat700-1	700	24.9%	11	9	**11**	**11**	5	0.47	10	249.4
p_hat700-2	700	49.8%	44	41	**44**	**44**	20	1.26	**44**	1022.3
p_hat700-3	700	74.8%	$\geq$ 62	59	60	**62**	29	1.76	**62**	1804.0
p_hat1000-1	1000	24.5%	$\geq$ 10	**10**	**10**	-	7	1.17	**10**	798.0
p_hat1000-2	1000	49.0%	$\geq$ 46	44	**46**	-	18	2.37	**46**	-
p_hat1000-3	1000	74.4%	$\geq$ 66	62	**65**	-	31	3.82	64	-
p_hat1500-1	1500	25.3%	12	10	11	-	9	3.12	**12**	-
p_hat1500-2	1500	50.6%	$\geq$ 65	**64**	63	-	28	7.69	**64**	-
p_hat1500-3	1500	75.4%	$\geq$ 94	91	**94**	-	43	11.43	91	-

by PBH. Note that here LDR performs remarkably well.

The test graphs prefixed with *keller* arise in conjunction with Keller's conjecture on tilings using hypercubes [10]. Here we could run PBH on only the two smallest instances due to memory restrictions. RD and PBH computed a maximum clique, and the latter also obtained the largest clique for the second instance.

Brockington and Culberson [9] developed their method that produced the graphs prefixed with *brock*. Their method uses a form of degree equalization to hide a large clique in a multitude of smaller ones. Also for this category PBH reached the largest cliques, except for one instance in which CBH found the maximum one. All other computed cliques are not maximum and the size gap between them and the maximum ones grows with the order of the graphs. The latter fact shows the effectiveness of Brockington and Culberson's approach for producing hard problems for algorithms based on the Motzkin–Straus continuous formulation.

The generation procedure for the Sanchis graphs (*san*) is described in [18, 29]. In 12 out of 15 cases, PBH produced the best results, and 11 of them were maximum cliques. In only one case did we obtain a clique smaller than that of AR, and in two cases RD performed slightly better. It is interesting to notice that AR and CBH obtained cliques that are, on average, half the size of those returned by PBH and RD.

Table 2
Performance of LDR, PBH, and other competing heuristics on unweighted DIMACS graphs (part II*). Entries that correspond to the best result for a given graph are boldfaced.*

Graph	Ord.	Dens.	ω	AR	CBH	RD	LDR	Time	PBH	Time
MANN_a9	45	92.7%	16	**16**	**16**	**16**	**16**	0.00	**16**	0.1
MANN_a27	378	99.0%	126	117	121	**125**	**125**	2.18	**125**	699.7
MANN_a45	1035	99.6%	345	-	336	-	340	43.72	**342**	-
keller4	171	64.9%	11	8	10	**11**	7	0.03	**11**	3.6
keller5	776	75.2%	27	16	21	25	15	1.27	**26**	1093.5
keller6	3361	81.8%	≥ 59	-	-	-	**31**	45.54	-	-
brock200_1	200	74.5%	21	19	**20**	**20**	13	0.07	**20**	9.7
brock200_2	200	49.6%	12	10	**12**	11	7	0.04	11	5.1
brock200_3	200	60.5%	15	13	**14**	**14**	10	0.6	**14**	6.4
brock200_4	200	65.8%	17	14	**16**	15	11	0.06	**16**	7.3
brock400_1	400	74.8%	27	20	23	**24**	17	0.37	**24**	111.6
brock400_2	400	74.9%	29	23	**24**	**24**	17	0.37	**24**	113.3
brock400_3	400	74.8%	31	23	23	**24**	17	0.37	**24**	111.2
brock400_4	400	74.9%	33	23	**24**	**24**	16	0.35	**24**	112.7
brock800_1	800	64.9%	23	18	20	**21**	13	1.18	**21**	858.6
brock800_2	800	65.1%	24	18	19	**20**	13	1.19	**20**	866.4
brock800_3	800	64.9%	25	19	**20**	**20**	15	1.34	**20**	864.5
brock800_4	800	65.0%	26	19	19	**20**	16	1.40	**20**	862.4
san200_0.7_1	200	70.0%	30	15	15	**30**	16	0.09	**30**	9.9
san200_0.7_2	200	70.0%	18	12	12	**18**	12	0.08	17	8.2
san200_0.9_1	200	90.0%	70	45	46	**70**	38	0.19	**70**	28.8
san200_0.9_2	200	90.0%	60	39	36	**60**	30	0.16	**60**	22.8
san200_0.9_3	200	90.0%	44	31	30	**44**	25	0.13	**44**	19.0
san400_0.5_1	400	50.0%	13	7	8	**13**	7	0.20	**13**	52.3
san400_0.7_1	400	70.0%	40	20	20	**40**	20	0.43	**40**	142.0
san400_0.7_2	400	70.0%	30	15	15	**30**	15	0.35	**30**	110.7
san400_0.7_3	400	70.0%	22	12	14	**19**	14	0.31	17	93.8
san400_0.9_1	400	90.0%	100	50	50	**100**	45	0.88	**100**	397.8
sanr200_0.7	200	69.7%	18	16	**18**	**18**	12	0.07	**18**	8.2
sanr200_0.9	200	89.8%	42	**41**	**41**	**41**	32	0.16	**41**	21.4
sanr400_0.5	400	50.1%	13	**13**	12	12	10	0.25	**13**	059.5
sanr400_0.7	400	70.0%	≥ 21	**21**	20	20	16	0.36	20	101.9
san1000	1000	50.2%	15	8	8	-	8	1.34	**15**	1185.0

Overall, these results show the clear superiority of PBH over both AR and CBH. It also turns out that PBH and RD perform equally well. However, the authors report in [20] that for graphs of order up to 500, the computational time of RD on a PC Pentium 166 MMX varied from 30 to 40 minutes on average, with a maximum of 1 h. 43 min. On larger graphs (i.e., up 800 vertices), the algorithm took from 17 min. to 8 h. 22 min. to converge. These high computational times prevented them from applying RD on graphs with more than 800 nodes.

Comparing complexity and computational times, however, is very difficult for this kind of heuristics. In fact we completely lack a clear complexity assessment of CBH, AR, and RD, and the computing times provided with each method refer to architectures and implementation solutions too different to be worth analyzing.

A remarkable empirical finding was that Algorithms 3.1 and 3.2 never failed to return a clique; hence, by Theorem 2.2, they always returned a maximal clique. Thus, we never needed to invoke any local search procedure in order to reach a nearby local minimizer. We tried to find exceptions by running them on random unweighted graphs with nonclique regular subgraphs, which do correspond to nonoptimal KKT points of (2). Hundreds of experiments were conducted on random instances with different degrees of noise, but they never failed to return a maximal clique. At the moment we cannot give a formal proof of this fact.

Before we present the results obtained by PBH on weighted graphs, it is worth discussing how it compares with other maximum clique heuristics that are *not* based on the Motzkin–Straus or related continuous formulations. Many such heuristics were presented at the second DIMACS implementation challenge on cliques, coloring, and satisfiability [19], and those based on tabu search, simulated annealing, and neural networks are among the most powerful. In the following discussion we shall neglect the easiest graph families, i.e., *c-fat*, *Hamming*, *Johnson*, and *MANN*, where straightforward greedy heuristics (and indeed Lemke's algorithm) already provide satisfactory results (see [30]).

In [30], Soriano and Gendreau presented three variants of tabu search for maximum cliques. The first two versions are deterministic algorithms. One uses a single tabu list of the last solutions visited, while the other uses an additional list (with an associated aspiration mechanism) containing the last vertices deleted. The third algorithm is probabilistic in nature and uses the same two tabu lists and aspiration mechanism as the second one. As it turns out, overall their results are comparable with those obtained with PBH. On the *p-hat* graphs, PBH obtained the same clique size as the three tabu search algorithms 8 times, it got smaller cliques in 6 cases (the difference being typically of one or two nodes), and in one case it yielded a larger clique. On the *keller* and the *brock* graphs, tabu search worked slightly better. In a few cases it obtained a larger clique, but when this happened the difference consisted of just a single vertex. Finally, on the *san* family the three tabu search heuristics did not perform equally well, the probabilistic one being the poorest. Here PBH obtained the same clique sizes as the double list variant in 11 cases, it returned a larger clique in 2 cases, and a smaller one twice. Compared to the single list heuristic, a similar picture emerges. Here PBH obtained a larger clique in three cases and a smaller one twice. It should be noticed that when PBH outperforms tabu search, the difference in clique size is significant (e.g., 30 vs. 19, 15 vs. 10, etc.), while the opposite is not true.

Homer and Peinado [16] compare three heuristics for maximum clique, namely, a straightforward greedy heuristic, a randomized version of Boppana and Halldórsson's subgraph-exclusion algorithm [8], and a version of simulated annealing with a simple cooling schedule. The algorithms were tested over very large graphs, and the overall conclusion was that simulated annealing outperforms the other competing algorithms. As far as comparison with PBH is concerned, it turns out that the average clique sizes obtained by simulated annealing in 1000 trials per graph on a selection of graphs from the *p-hat*, *keller*, and *brock* families (no results are presented on the Sanchis graphs) are always rather smaller than those obtained by PBH, which, by contrast, is run only once. There is only one exception: the *p_hat*1500-3 graph, where PBH found a clique of 91 vertices and the average clique size found by simulated annealing was 92.2. Looking at the best results obtained over the 1000 runs, it turns out that simulated annealing equaled PBH 8 times, found a slightly larger clique in another 8 cases (usually one vertex larger, except for *p_hat*1500-3), and in a single case PBH got a better result.

In [18], Jagota, Sanchis, and Ganesan developed several neural-network heuristics based on the so-called Hopfield model to approximate maximum clique. Overall, the best results were obtained using a *greedy steep descent* (GSD) dynamics, although it was slower than the others. The best results on the Sanchis graphs, in contrast, were obtained using a stochastic steep descent heuristic endowed with a "reinforcement learning" strategy that automatically adjusts the internal parameters as the process evolves (SSD_{RL}). PBH significantly outperforms these models. Specifically, on the *brock*, *keller*, and *p-hat* graphs, PBH always found cliques of size larger than or equal to those found by GSD. On the *san* family the contrast is even more evident. Here PBH found a clique larger than SSD_{RL} 11 times and obtained the same clique size in the remaining 4 cases. In a few cases, the cliques found by PBH were substantially larger than those found by SSD_{RL} (44 vs. 33, 30 vs. 18, etc.).

Grossman [14] also proposed a neural-network heuristic based on the Hopfield model, originally designed for an all-optical implementation. The model has a threshold parameter which determines the character of the stable states of the network. The author suggests an annealing strategy on this parameter, and an adaptive procedure to choose the network's initial state and threshold. Experiments over random as well as selected DIMACS graphs are reported. (Being a randomized procedure, for each graph hundreds of trials were performed.) Compared to PBH, a picture similar to simulated annealing emerges. The average clique sizes found by Grossman's heuristic are substantially smaller than those returned by PBH on all graph families. (No results on the Sanchis family are presented in [14].) Taking the best results found, out of 17 instances PBH found a larger clique in 5 cases, a smaller one in 4 cases, and the same clique size in the remaining 8 instances. Again, we stress the fact that PBH is run only once on each graph instance and no randomization takes place.

4.2. Weighted graphs. For the weighted case there are no widely accepted benchmark graphs, and therefore we adopted weighted random graphs as a testbed for Algorithm 3.3. To obtain the weighted clique number for each test graph, we used Babel's method [1], which is one of the most efficient algorithms available in the MWCP literature. Babel uses a branch and bound approach as follows: Upper and lower bounds for the maximum weight clique are found by coloring the weighted graph, where the number of colors represents the total sum of all weights. The branching part of Babel's algorithm divides the bounded search-tree into smaller subproblems, the branching decisions depending on a specific order of all possible remaining nodes. By applying these steps recursively, the maximum weight clique will be found in finite time, and for not too big and too dense graphs in very short time. Unfortunately, the coloring heuristic employed by this method severely restricts node weights to discrete values. For example, if we consider graphs with floating point weights between 1 and 10, and with 3 significant digits, this would lead to as many as 9,000 possible discrete weights. This means that Babel's method could use up to 900,000 colors in a graph of order 100. To accommodate this deficiency we generated random graphs with random integer weights ranging between 1 and 10.

In this series of experiments we did not run the LDR algorithm, because of the poor performance obtained on unweighted graphs. Given the clique C found by Algorithm 3.3, as a success measure we took the ratio $R = W(C)/\omega(G, w)$. Table 3 lists average results (*Avg. R* columns) and their standard deviations (*St. Dev.* columns) for families of 20 random graphs with 100 vertices and various density values p.

Usual random graphs (*Normal* in Table 3) tend to be very regular (i.e., the degree of all nodes is nearly the same). This feature is typically not shared by real-world instances; hence we used Algorithm 4.1, borrowed from [7], to generate more irregular instances (*Irregular*). The same intent drove the choice of performing tests over families of DIMACS *p-hat* graphs (*p-hat* columns).

On all types of graphs we obtained very positive figures. In particular, for normal

Algorithm 4.1. An edge generation procedure for random irregular graphs.

```
$\mu = p\binom{n}{2}$.
while ($\mu > 0$)
    choose randomly $v \in \{1, \dots, n\}$ and $d \in \{1, \dots, n-1\}$;
    add $d$ edges to randomly chosen neighbors of $v$;
    if this is not possible
        add the maximum of free edges to neighbors of $v$;
    $\mu = \mu -$ number of actually added edges.
endwhile;
```

Table 3

Performance of Algorithm 3.3 *on weighted random graphs with* 100 *vertices (see text for explanation).*

p	Normal		Irregular		p-hat	
	Avg. R	St. Dev.	Avg. R	St. Dev.	Avg. R	St. Dev.
0.10	97.95%	±0.15	98.44%	±0.13	99.33%	±0.09
0.20	97.73%	±0.16	98.63%	±0.12	97.17%	±0.17
0.30	97.25%	±0.17	98.84%	±0.11	96.38%	±0.20
0.40	95.04%	±0.23	98.53%	±0.12	97.54%	±0.16
0.50	94.61%	±0.24	98.74%	±0.12	94.56%	±0.24
0.60	94.71%	±0.23	99.64%	±0.06	96.20%	±0.20
0.70	96.10%	±0.20	98.94%	±0.11	94.44%	±0.24
0.80	93.13%	±0.26	98.56%	±0.12	94.64%	±0.23
0.90	94.29%	±0.24	99.56%	±0.07	95.26%	±0.22
0.95	96.49%	±0.19	99.75%	±0.05	94.49%	±0.24

random graphs one can see how efficiency slowly decreases with increasing density but always remains above 93%. For irregular graphs these figures improve considerably, never falling below 98.4% efficiency. The same can be said for the *p-hat* graphs. But in this last case it must be taken into account that for p close to 0.5, the node degree variance is largest. The table reflects this fact in that performance is optimal for sparse graphs, is worst for p close to 0.5, and then slowly improves while moving toward $p = 1$. At this end-point the increased density becomes the dominant reason for not reaching the heaviest clique.

The above experiments were conducted on a machine equipped with a 400MHz Alpha CPU. On this machine, computing times for PBH ranged (approximately) between 0.6 and 9 seconds.

5. Conclusions. We have presented an effective heuristic for the MWCP which employs a pivoting algorithm on an LCP problem formulation derived from a development of the Motzkin–Straus theorem. The remarkable effectiveness of our approach and the empirical immunity of Lemke's method to saddle points seems to indicate that pivoting-based methods offer a promising new way to tackle this and related combinatorial problems. The algorithm has already been applied with success to graph matching problems arising in computer vision and pattern recognition [21]. Note also that our algorithm is completely devoid of working parameters, a valuable feature which distinguishes it from other heuristics proposed in the literature (see, e.g., [4]). In future investigations we will try to give a formal proof of convergence to local minimizers, and we will tackle the problem of reducing the time and space complexity of our method.

Appendix. Optimality and strict complementarity. Here we provide a discussion of second-order optimality conditions for (2), where $f(x) = x^TQx$ with a general symmetric $n \times n$ matrix rather than Q_G, in relation to the strict complementarity condition. First we rephrase optimality in terms of copositivity with respect to a polyhedral cone Γ. Recall that, given a cone $\Gamma \subseteq \mathbb{R}^n$, a symmetric matrix Q is said to be Γ-*copositive* if $x^TQx \geq 0$ for all $x \in \Gamma$. If the inequality holds strictly for all $x \in \Gamma \setminus \{o\}$, then Q is said to be *strictly* Γ-*copositive.* As usual, define $e^\perp = \{v \in \mathbb{R}^n : e^Tv = 0\}$.

THEOREM A.1. *Let $x \in \Delta$ and $\gamma = x^TQx$. If x is a KKT point of* (2), *then set*

$$\Gamma^\star(x) = \{v \in e^\perp : v_i \geq 0 \text{ if } i \in V \setminus S(x) \text{ and } v^TQx = 0\}\,. \tag{11}$$

Then

(a) *x is a local solution to* (2) *if and only if Q is $\Gamma^\star(x)$-copositive;*

(b) *x is a strict local solution to* (2) *if and only if Q is strictly $\Gamma^\star(x)$-copositive.*

Proof. If $\Gamma(x) = \{v \in \mathbb{R}^n : v_i \geq 0 \text{ if } i \in V \setminus S(x)\}$ denotes the tangent cone of Δ at x, then $\Gamma^\star(x)$ as defined in (11) satisfies $\Gamma^\star(x) = \{v \in \Gamma(x) : v^T\nabla f(x) = 0\}$, i.e., coincides with the reduced tangent cone. Hence (a) is established by Theorem 2 of [2], while (b) can be proved by a simpler variant of the argument therein. ☐

Observe that, in light of the above conditions, the last statement of Theorem 2.1 can be rephrased as follows: If Q_G is $\Gamma^\star(x)$-copositive, then Q_G is even strictly so. (This holds also for every other matrix $Q \in \mathcal{C}(G, w)$, the entire class introduced in [3].)

For quadratic problems over polyhedra more general than Δ, there are similar second-order optimality conditions, also for global optimality; see, e.g., [2]. All conditions involve checking copositivity, which from a practical point of view should be avoided, as checking copositivity is NP-hard [25] whereas checking definiteness (see below) can be done in polynomial time. In contrast with several other problems (e.g., the simplex method in linear optimization), this difference in worst-case complexity is also reflected in the actual average case behavior. Thus an additional aspect of the significance of strict complementarity becomes evident.

THEOREM A.2. *If $x \in \Delta$ is a KKT point of* (2) *which satisfies the strict complementarity condition, then the reduced tangent cone*

$$\Gamma^\star(x) = \{v \in e^\perp : v_i = 0 \text{ if } i \in V \setminus S(x)\} \tag{12}$$

becomes a linear subspace.

Further, if x is a vertex of Δ, then x is a strict local solution to (2).

Otherwise, assume that x has $r + 1 \geq 2$ strictly positive coordinates, pick a fixed $i \in S(x)$, and form the symmetric $r \times r$ matrix

$$\bar{Q} = [q_{ii} + q_{jk} - q_{ij} - q_{ik}]_{(j,k)\in S(x)\setminus\{i\}\times S(x)\setminus\{i\}}\,. \tag{13}$$

Then

(a) *x is a strict local solution to* (2) *if and only if $\bar{Q}$ is positive-definite;*

(b) *x is a local solution to* (2) *if and only if $\bar{Q}$ is positive-semidefinite.*

Proof. To show (12), we employ the KKT conditions (4). Then $\lambda_i > 0$ for all $i \in V \setminus S(x)$ implies via (11) and

$$0 = v^T\nabla f(x) = \sum_{i\in V\setminus S(x)} \lambda_i v_i - \lambda e^T v = \sum_{i\in V\setminus S(x)} \lambda_i v_i$$

that $v_i = 0$ for all $i \in V \setminus S(x)$ if $v \in \Gamma^\star(x)$. The converse is also obvious. Next, if x is a vertex of the feasible set, then $\Gamma^\star(x) = \{o\}$, so that the copositivity condition in Theorem A.1(a) is void. (Note that local optimality of vertices which are strictly complementary KKT points holds in a much more general context.) Now assume that x is no vertex. Denoting again by e_i the ith column of the $n \times n$ identity matrix I_n, we obtain a basis for $\Gamma^\star(x)$ by $\{e_i - e_j : j \in S(x) \setminus \{i\}\}$ and collect these vectors as columns of an $n \times r$ matrix U so that $\Gamma^\star(x) = U(\mathbb{R}^r)$. But U can be written, after suitable reordering, as $U = [e, -I_r, O]^T$. Now partition Q into appropriate blocks to arrive at $\bar{Q} = U^T Q U$ as in (13). As a consequence, Q is $\Gamma^\star(x)$-copositive if and only if $\bar{Q}$ is positive-semidefinite, and similarly for the strict versions. □

A consequence of the last statement in Theorem 2.1 is that, under strict complementarity, the (positive-semidefinite) matrices $\bar{Q}$ are nonsingular if $Q = Q_G$. (Again, this holds for every $Q \in \mathcal{C}(G, w)$, the entire class introduced in [3].) Thus Theorem A.2 can be viewed as a sort of converse of Theorem 2.4, where nonsingularity of certain matrices in turn guarantees strict complementarity.

Acknowledgments. The authors wish to thank Richard Cottle for the valuable assistance he was willing to provide during early and late stages of this work, and the anonymous referees for their helpful comments. M.P. would also like to thank Steven Zucker for stimulating discussions which triggered this research.

REFERENCES

[1] L. Babel, *A fast algorithm for the maximum weight clique problem*, Computing, 52 (1994), pp. 31–38.

[2] I. M. Bomze, *Copositivity conditions for global optimality in indefinite quadratic programming problems*, Czech. J. Oper. Res., 1 (1992), pp. 7–19.

[3] I. M. Bomze, *On standard quadratic optimization problems*, J. Global Optim., 13 (1998), pp. 369–387.

[4] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, *The maximum clique problem*, in Handbook of Combinatorial Optimization—Suppl. Vol. A, D.-Z. Du and P. M. Pardalos, eds., Kluwer Academic Publishers, Boston, MA, 1999, pp. 1–74.

[5] I. M. Bomze, M. Budinich, M. Pelillo, and C. Rossi, *Annealed replication: A new heuristic for the maximum clique problem*, Discrete Appl. Math., 2002, to appear.

[6] I. M. Bomze, M. Pelillo, and R. Giacomini, *Evolutionary approach to the maximum clique problem: Empirical evidence on a larger scale*, in Developments in Global Optimization, I. M. Bomze, T. Csendes, R. Horst, and P. M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp. 95–108.

[7] I. M. Bomze, M. Pelillo, and V. Stix, *Approximating the maximum weight clique using replicator dynamics*, IEEE Trans. Neural Networks, 11 (2000), pp. 1228–1241.

[8] R. Boppana and M. Halldórsson, *Approximating maximum independent sets by excluding subgraphs*, BIT, 32 (1992), pp. 180–196.

[9] M. Brockington and J. C. Culberson, *Camouflaging independent sets in quasi-random graphs*, in Cliques, Coloring and Satisfiability, D. Johnson and M. A. Trick, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 26, AMS, Providence, RI, 1996, pp. 75–88.

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

[11] R. W. Cottle, J. Pang, and R. E. Stone, *The Linear Complementarity Problem*, Academic Press, Boston, MA, 1992.

[12] L. E. Gibbons, D. W. Hearn, and P. M. Pardalos, *A continuous-based heuristic for the maximum clique problem*, in Cliques, Coloring and Satisfiability, D. Johnson and M. A. Trick, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 26, AMS, Providence, RI, 1996, pp. 103–124.

[13] L. E. Gibbons, D. W. Hearn, P. M. Pardalos, and M. V. Ramana, *Continuous characterizations of the maximum clique problem*, Math. Oper. Res., 22 (1997), pp. 754–768.

[14] T. Grossman, *Applying the INN model to the maximum clique problem*, in Cliques, Coloring and Satisfiability, D. Johnson and M. A. Trick, eds., DIMACS Ser. Discrete Math. Theoret.

Comput. Sci. 26, AMS, Providence, RI, 1996, pp. 125–145.

[15] J. Hofbauer and K. Sigmund, *Evolutionary Games and Population Dynamics*, Cambridge University Press, Cambridge, UK, 1998.

[16] S. Homer and M. Peinado, *Experiments with polynomial-time CLIQUE approximation algorithms on very large graphs*, in Cliques, Coloring and Satisfiability, D. Johnson and M. A. Trick, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 26, AMS, Providence, RI, 1996, pp. 147–167.

[17] R. Horaud and T. Skordas, *Stereo correspondence through feature grouping and maximal cliques*, IEEE Trans. Pattern Anal. Machine Intell., 11 (1989), pp. 1168–1180.

[18] A. Jagota, L. Sanchis, and R. Ganesan, *Approximately solving maximum clique using neural network and related heuristics*, in Cliques, Coloring and Satisfiability, D. Johnson and M. A. Trick, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 26, AMS, Providence, RI, 1996, pp. 169–204.

[19] D. Johnson and M. A. Trick, eds., *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 26, AMS, Providence, RI, 1996.

[20] A. Kuznetsova and A. Strekalovsky, *On solving the maximum clique problem*, J. Global Optim., 21 (2001), pp. 265–288.

[21] A. Massaro and M. Pelillo, *A complementary pivoting approach to graph matching*, in Energy Minimization Methods in Computer Vision and Pattern Recognition, M. Figueiredo, J. Zerubia, and A. K. Jain, eds., Lecture Notes in Comput. Sci. 2134, Springer-Verlag, Berlin, 2001, pp. 469–479.

[22] A. Menon, K. Mehrotra, C. K. Mohan, and S. Ranka, *Optimization using replicators*, in Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburg, PA, 1995, Morgan Kaufmann, San Francisco, CA, pp. 209–216.

[23] D. A. Miller and S. W. Zucker, *Efficient simplex-like methods for equilibria of nonsymmetric analog networks*, Neural Computation, 4 (1992), pp. 167–190.

[24] T. S. Motzkin and E. G. Straus, *Maxima for graphs and a new proof of a theorem of Turán*, Canad. J. Math., 17 (1965), pp. 533–540.

[25] K. G. Murty and S. N. Kabadi, *Some NP-complete problems in quadratic and linear programming*, Math. Programming, 39 (1987), pp. 117–129.

[26] M. Pelillo, *Replicator dynamics in combinatorial optimization*, in Encyclopedia of Optimization, C. A. Floudas and P. M. Pardalos, eds., Kluwer Academic Publishers, Boston, MA, 2001.

[27] M. Pelillo and A. Jagota, *Feasible and infeasible maxima in a quadratic program for the maximum clique problem*, J. Artificial Neural Networks, 2 (1995), pp. 411–420.

[28] M. Pelillo, K. Siddiqi, and S. W. Zucker, *Matching hierarchical structures using association graphs*, IEEE Trans. Pattern Anal. Machine Intell., 21 (1999), pp. 1105–1120.

[29] L. Sanchis, *Test case construction for the vertex cover problem*, in Computational Support for Discrete Mathematics, N. Dean and G. E. Shannon, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 15, AMS, Providence, RI, 1994, pp. 315–326.

[30] P. Soriano and M. Gendreau, *Tabu search algorithms for the maximum clique problem*, in Cliques, Coloring and Satisfiability, D. Johnson and M. A. Trick, eds., DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 26, AMS, Providence, RI, 1996, pp. 221–242.