SWAPS, DIVERSIFICATION, AND THE COMBINATORICS OF PIVOTING FOR THE MAXIMUM WEIGHT CLIQUE

MARCO LOCATELLI*, IMMANUEL M. BOMZE†, AND MARCELO PELILLO‡

Abstract. Recently a powerful pivoting-based heuristic (PBH) has been introduced for attacking the maximum weight clique problem, based on the linear complementarity formulation of an equivalent (standard) quadratic program. In this paper, we show that PBH is equivalent to a combinatorial greedy heuristic. This interpretation allows us to modify PBH, by introducing swap and diversification strategies, particularly focusing on the unweighted case. Results over standard DIMACS graphs show that the resulting heuristic compares well with the most powerful algorithms available in the literature: results are considerably improved while computation time is kept low.

Key words. maximum clique, linear complementarity, pivoting methods, greedy heuristics, combinatorial optimization

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

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 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 in general, but that MCP coincides with MWCP in the special case when the weights assigned to the vertices are all equal). 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. We refer to [3] for a recent review concerning algorithms, applications and complexity issues of this important problem.

Motivated by a recent quadratic programming formulation, which generalizes an earlier remarkable result by Motzkin and Straus, in a recent paper [11] a new pivoting-based heuristic (PBH) for the MWCP has been proposed which is based on the corresponding linear complementarity problem (LCP). The algorithm is essentially a variant of Lemke’s classical algorithm that incorporates an effective look-ahead pivot rule, and it proved to be among the most powerful MWCP heuristics available in the literature.

The objective of this paper is twofold. First, after briefly describing PBH we provide some theoretical results which prove a few conjectures that have been put forward in [11] based on empirical observations. Interestingly, these results allow for an interesting combinatorial interpretation of the algorithm: PBH is essentially equivalent to a greedy combinatorial heuristic. Motivated by these results, we then focus on the unweighted case and modify the combinatorial counterpart of PBH to

* Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, I-10149 Torino, Italy (locatelli@di.unito.it)
† Institut für Statistik und Decision Support Systems, Universität Wien, Universitätsstraße 5, A-1010 Wien, Austria (immanuel.bomze@univie.ac.at)
‡ Dipartimento di Informatica, Università Ca’ Foscari di Venezia, Via Torino 155, I-30127 Venezia Mestre, Italy (pelillo@dsi.unive.it).
improve the quality of the results obtained, at the cost of slightly increasing the computation time. Basically, the major modification incorporates vertex swaps during the clique construction process, but we also equip the algorithm with a simple and effective diversification strategy. The resulting algorithms have been tested extensively on various instances of DIMACS benchmark graphs and the results obtained clearly show the effectiveness of the approach.

2. The pivoting-based heuristic and its combinatorial interpretation.

2.1. Notations and definitions. Let \( G = (V, E, w) \) be an arbitrary undirected and weighted graph, where \( V = \{1, \ldots, n\} \) is the vertex set, \( \binom{V}{2} \) denotes the system of all two-element subsets of \( V \), \( E \subseteq \binom{V}{2} \) is the edge set and \( w \in \mathbb{R}^n \) is the weight vector, the \( i \)-th 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 \( d(i) = |N_i| \), 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 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 \( G \).

2.2. The PBH heuristic. Given a weighted graph \( G = (V, E, w) \), consider the following standard quadratic program (StQP):

\[
\begin{align*}
\text{minimize} & \quad x^T Q_G x \\
\text{subject to} & \quad x \in \Delta
\end{align*}
\]

where the matrix \( Q_G = (q_{ij})_{(i,j) \in V \times V} \) is defined as:

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

and \( \Delta \) denotes the standard simplex in the \( n \)-dimensional Euclidean space \( \mathbb{R}^n \):

\[
\Delta = \left\{ x \in \mathbb{R}^n : \sum_{i \in V} x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i \in V \right\} .
\]
Given a subset of vertices $S \subseteq V$, the \textit{weighted characteristic vector} of $S$, denoted by $x^{S,w}$, is the vector in $\Delta$ whose coordinates are given by:

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

The PBH heuristic is based upon the following result [4].

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

- A vector $x \in \Delta$ is a local solution of (1) if and only if $x = x^{S,w}$, where $S$ is a maximum clique of $G$.
- A vector $x \in \Delta$ is a global solution of (1) if and only if $x = x^{S,w}$, where $S$ is a maximum weight clique of $G$.

Moreover, all solutions of (1) are strict and are characteristic vectors of maximal cliques of $G$.

It is well known that stationary points of quadratic optimization problems with linear constraints 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 [8]. Hence, once that 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 [11].

Specifically, computing the stationary points of (1) can be done by solving the LCP $(q_G,M_G)$, which is the problem of finding a vector $x$ satisfying the system

$$y = q_G + M_G x \geq 0, \quad x \geq 0, \quad x^T y = 0$$

where

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

with $Q_G$ as in (2) and $e$ is the vector with all coordinates equal to 1. With the above definitions, it is well known that if $x \in \mathbb{R}^n$ solves (3) - we say then that $z = [x^T,y^T]^T$ is a complementary solution of LCP $(q_G,M_G)$ - then $x$ is a stationary point of (1). Note that $Q_G$ is strictly $\mathbb{R}^n_+$-co-positive, hence so is $M_G$ and this is sufficient to ensure that LCP $(q_G,M_G)$, or (3), always has a solution [8].

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 largely for its ability to provide a solution for several matrix classes. Given the generic LCP $(q,M)$, it deals with the augmented problem $(q,d,M)$ defined by

$$y = q + [M,d] \begin{bmatrix} x \\ \theta \end{bmatrix} \geq 0, \quad \theta \geq 0, \quad x \geq 0, \quad x^T y = 0.$$
Unfortunately, like other pivoting schemes, the convergence of Lemke’s algorithm is guaranteed only for non-degenerate problems, and ours is indeed degenerate. In [11], standard degeneracy resolution strategies were tested over a number of 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 this property is exploited to develop a variant of Lemke’s algorithm: the look-ahead pivot rule which we will now shortly describe, for the readers’ convenience.

As customary, we will use a superscript for the problem data and, to simplify notation, subscripts indicating the dependence on graph $G$ will be omitted. Hence, $q^\nu$ and $M^\nu$ will identify the situation after $\nu$ pivots and $Q^\nu$ will indicate the leading principal $n \times n$ submatrix of $M^\nu$. Consistently, $y^\nu$ and $x^\nu$ will indicate the vectors of basic and non-basic variables, respectively, each made up of a combination of the original $x_i$ and $y_i$ variables. The notation $(x^\nu_i, y^\nu_i)$ will be used to indicate pivoting transformations. The index set of the basic variables that satisfy the min-ratio test at iteration $\nu$ will be denoted with $\Omega^\nu$, i.e.

$$\Omega^\nu = \arg\min_i \left\{ \frac{-q^\nu_i}{m^\nu_{i,s}} : m^\nu_{i,s} < 0 \right\}$$

where $s$ is the index of the driving column. Also, in the sequel the auxiliary column that contains the covering vector $d$ in (5) will be referred to as the column $n + 3$ of matrix $M = M_G$. The non-degeneracy assumption basically amounts to having $|\Omega^\nu| = 1$ for all $\nu$, thereby excluding any cycling behavior.

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

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

where $\Omega^\nu_{i+1}$ is the index set of those variables that would satisfy the min-ratio test at iteration $\nu + 1$ if the driving variable at iteration $\nu$ were blocked with $y^\nu_i$ 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 refer to this rule as the look-ahead (pivot) rule. The resulting procedure is specified as Algorithm 2.1 below.

Empirical evidence indicated $h$ as a key parameter for the quality of the final result of Algorithm 2.1. Unfortunately no effective means could be identified to restrict the choice of values in $V$ that can guarantee a good sub-optimal solution, so one has to consider iterating for most, if not all, vertices of $V$ as outlined in Algorithm 2.2. A simple criterion avoids considering those nodes that cannot drive to larger cliques than the current one, because their weight and that of their neighborhood is too small. Unfortunately, this criterion is effective only for very sparse graphs. It has been observed that the schema is sensitive to the ordering of nodes. Since the best figures were obtained by reordering $G$ by decreasing weight of each node and its neighborhood, this feature is formalized in Algorithm 2.2, which is the Pivoting Based Heuristic (PBH).
Algorithm 2.1  A reduced version of Lemke’s Scheme I with the look-ahead rule, applied to the MWCP.

**Input:** A graph $G = (V, E, w)$ and $h \in V$.

Let $Q_h = (q_{ij})$, $\nu \leftarrow 0$, $K \leftarrow \emptyset$.

The driving variable is $x_h$.

Infinite loop

- Let $x^*_h$ denote the driving variable.
- $\Omega^* = \{ i : q_{i_h} > 0 \}$.
- If $\Omega^* \subseteq \{ h \}$ stop: the result is $K$.
- $\Phi^* = \arg \min_i \{ |\Omega^i| - |\Omega^{i+1}| > 0 : i \in \Omega^* \}$.
- $r = \min \Phi^*$
- If $y^*_i \equiv x_i$ for some $i$, then $K \leftarrow K \setminus \{ i \}$
- Perform $(y^*_i, x^*_i)$
- The new driving variable is the variable complementary to $y^*_i$
- $\nu \leftarrow \nu + 1$
- If $y^*_i \equiv x_i$ for some $i$, then $K \leftarrow K \cup \{ i \}$

Algorithm 2.2  The pivoting-based heuristic (PBH) for the MWCP.

**Input:** A graph $G = (V, E, w)$.

Let $G' = (V', E', u')$ be a permutation of $G$

with $W \{ \{ u \} \cup N_u \} = W \{ \{ u' \} \cup N_{u'} \}$ for all $u, u' \in V'$ with $u < u'$.

$K^* \leftarrow \emptyset$.

For $v' = 1, \ldots, n : W \{ \{ v \} \cup N_{v} \} > W (K^*)$ do

- Run Algorithm 2.1 with $G'$ and $v'$ as input.
- Let $K$ be the obtained result.
- If $W (K) > W (K^*)$, then $K^* \leftarrow K$.

The result is the mapping of $K^*$ in $G$.

2.3. Combinatorial interpretation of PBH. Given a set $S \subseteq V$ of vertices, abbreviate by $S_i = S \setminus N_i$ the vertices in $S$ that are not adjacent to $i$, and denote by

$$d_S(i) = \sum_{j \in S_i} \left(1 + \frac{w_j}{w_i}\right), \quad i \in V \setminus S$$

(twice) the average of weighted and unweighted co-degree of $i$ w.r.t. $S$. Note that if $S_i \neq \emptyset$, then $d_S(i) \geq 1$, and that additivity holds: two disjoint subsets $S$ and $T$ satisfy $d_{S \cup T}(i) = d_S(i) + d_T(i)$ for any $i \in V \setminus (S \cup T)$. Further, for $i \neq j$ we have

$$\frac{1}{tw_j} \Delta_{ij}(i) = a_{ij}$$

as defined in (2).

Let us represent the tableau at iteration $\nu$ of Algorithm 2.1 in Table 1, where $x_p$ denotes the driving variable. Notice that, without loss of generality, the vertices of the graph have been numbered in such a way that the variables which entered the basis up to iteration $\nu$ are those corresponding to vertices 1 up to $p-1$; these variables entered the basis in the same order of the vertices (i.e. $x_1$ first, then $x_2$, and so on); the driving variable $x_p$ is the one corresponding to vertex $p$. Note that each entry $a^*_{i,j}$
Table 1

Tableau at iteration $\nu$ of Algorithm 2.1

<table>
<thead>
<tr>
<th>$q$</th>
<th>$y_2$</th>
<th>$\ldots$</th>
<th>$y_p$</th>
<th>$x_p$</th>
<th>$\ldots$</th>
<th>$x_n$</th>
<th>$y_1$</th>
<th>$x_{n+2}$</th>
<th>$y_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{n+1}$</td>
<td>1</td>
<td>$a_{1,1}^p$</td>
<td>$\ldots$</td>
<td>$a_{1,p-1}^p$</td>
<td>$a_{1,p}^p$</td>
<td>$a_{1,n}^p$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>$a_{2,1}^p$</td>
<td>$\ldots$</td>
<td>$a_{2,p-1}^p$</td>
<td>$a_{2,p}^p$</td>
<td>$a_{2,n}^p$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_{p-1}$</td>
<td>0</td>
<td>$a_{p,1}^p$</td>
<td>$\ldots$</td>
<td>$a_{p,p-1}^p$</td>
<td>$a_{p,p}^p$</td>
<td>$a_{p,n}^p$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_{p+1}$</td>
<td>0</td>
<td>$a_{p+1,1}^p$</td>
<td>$\ldots$</td>
<td>$a_{p+1,p-1}^p$</td>
<td>$a_{p+1,p}^p$</td>
<td>$a_{p+1,n}^p$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$y_n$</td>
<td>0</td>
<td>$a_{n,1}^p$</td>
<td>$\ldots$</td>
<td>$a_{n,n-1}^p$</td>
<td>$a_{n,n}^p$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\theta$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

of the tableau corresponds to the following couple of variables

$$(x_{i-1}, x_j) \quad \text{for } i = 1, \ldots, p, \quad \text{and } j = p, \ldots, n;$$

$$(y_i, x_j) \quad \text{for } i = p + 1, \ldots, n, \quad \text{and } j = p, \ldots, n;$$

$$(x_{i-1}, y_{j+1}) \quad \text{for } i = 1, \ldots, p, \quad \text{and } j = 1, \ldots, p - 1;$$

$$(y_i, y_{j+1}) \quad \text{for } i = p + 1, \ldots, n, \quad \text{and } j = 1, \ldots, p - 1.$$

where $x_0 \equiv x_{n+1}$.

Given the situation displayed in the tableau we now prove the following results, which allow us to give a nice combinatorial interpretation of PBH and to prove a few conjectures stated in [11].

**Proposition 2.2.** At iteration $\nu$ of the PBH algorithm, let $x_p$ be the driving variable and $K = \{i \in V : x_i \text{ is basic} \} = \{1, \ldots, p - 1\}$. Then, the following statements are true:

1. $\forall i \notin K \cup \{p\}$

$$a_{i,p}^p = \frac{1}{2w_p} \left[ \hat{d}_{K \cup \{p\}}(i) - 1 \right].$$

which can be negative only if $a_{i,p}^p = -\frac{1}{2w_p}$, i.e. when $K \cup \{p\} \setminus N_i = \emptyset$ or, equivalently, vertex $i$ is adjacent to any vertex in $K \cup \{p\}$.

2. $\forall i \notin K \cup \{p\}$ such that $a_{i,p}^p < 0$ (or equivalently, in view of point 1., $a_{i,p}^p = -\frac{1}{2w_p}$), it holds that

$$a_{i,i}^p = \frac{1}{2w_i}.$$

3. $\forall i \notin K \cup \{p\}$ such that $a_{i,p}^p < 0$ it holds that $\forall s \notin K \cup \{i,p\}$

$$a_{s,i}^p = \frac{1}{2w_i} \hat{d}_{(i)}(s) = q_{is}.$$

**Proof.** The proof is by induction on $\nu$. We first prove that the result is true at iteration $\nu = 1$ and then we assume that it is true at iteration $\nu \geq 1$ and we prove it is true at iteration $\nu + 1.$
\( \nu = 1 \)

**Point 1** For any \( i \neq p \) it holds that

\[
a_{i,p}^1 = q_{ip} - q_{pp} = q_{ip} - \frac{1}{2w_p}
\]

Since

\[
q_p = \begin{cases} 
0 & \text{if } \{i, p\} \in E \\
\frac{1}{2w_i} + \frac{1}{2w_p} & \text{otherwise}
\end{cases} = \frac{1}{2w_p} \hat{d}_{K \cup \{p\}}(i)
\]

for \( K = \emptyset \), point 1 holds.

**Point 2** It follows from (6) and (7) that \( a_{i,p}^1 < 0 \) implies \( q_{ip} = q_{pi} = 0 \). Since

\[
a_{i,i}^1 = q_{ii} - q_{pi} = \frac{1}{2w_i} - 0 = \frac{1}{2w_i}
\]

point 2 follows.

**Point 3** Now let \( a_{i,p}^1 < 0 \). For any \( s \neq i, p \)

\[
a_{s,i}^1 = q_{si} - q_{pi} = q_{si} = \frac{1}{2w_i} \hat{d}_{(i)}(s)
\]

so that point 3 is satisfied.

**From \( \nu \) to \( \nu + 1 \)**

Again by renumbering of the vertices if necessary we can assume that the new driving variable at iteration \( \nu + 1 \) is \( x_{p+1} \). Then, in particular, it holds that \( a_{p+1,p}^\nu < 0 \) or, equivalently, in view of point 1 of the inductive assumption, \( a_{p+1,p}^\nu = -\frac{1}{2w_p} \). For any \( s, t, s \neq p+1, t \neq p \), it therefore holds, in view of the pivoting update, that

\[
a_{s,t}^{\nu+1} = a_{s,t}^\nu - \frac{a_{p+1,t}^\nu a_{s,p}^\nu}{a_{p+1,p}^\nu} = a_{s,t}^\nu + 2w_p a_{p+1,t}^\nu a_{s,p}^\nu,
\]

where the last equality follows from point 1 of the inductive assumption (put \( i = p+1 \)).

Next, for \( t \neq p \)

\[
a_{p+1,t}^{\nu+1} = \frac{a_{p+1,t}^\nu}{a_{p+1,p}^\nu} = 2w_p a_{p+1,t}^\nu.
\]

Finally,

\[
a_{p+1,p}^{\nu+1} = \frac{1}{a_{p+1,p}^\nu} = -2w_p.
\]

**Point 1** For any \( s \not\in K \cup \{p, p+1\} \) it follows from (8) for \( t = p+1 \) that

\[
a_{s,p+1}^{\nu+1} = a_{s,p+1}^\nu + 2w_p a_{p+1,p}^\nu a_{s,p}^\nu = a_{s,p+1}^\nu + \frac{1}{2w_{p+1}} \left( \hat{d}_{K \cup \{p\}}(s) - 1 \right),
\]

where the last equality follows from point 1 (put \( i = s \)) and point 2 (put \( i = p + 1 \)) of the inductive assumption. In view of point 3 of the inductive assumption \((i = p + 1)\), we have

\[
a_{s,p+1}^\nu = \frac{1}{2w_{p+1}} \hat{d}_{(p+1)}(s),
\]
and therefore, using the additivity property for \( \tilde{d} \) addressed above,

\[
a_{s,p+1}^{\nu+1} = \frac{1}{2w_{p+1}} \left[ \tilde{d}_{K \cup \{p,p+1\}}(s) - 1 \right],
\]

which means that point 1 holds also at iteration \( \nu + 1 \).

**Point 2** Now let us assume that \( a_{i,p+1}^{\nu+1} < 0 \). We want to prove that \( a_{i,i}^{\nu+1} = \frac{1}{2w_i} \). First we notice that in view of the proof of point 1 it holds that \( a_{i,p+1}^{\nu+1} < 0 \) if and only if

\[
\begin{align*}
  (11) & \quad a_{i,p+1}^{\nu} = 0; \\
  (12) & \quad a_{i,p}^{\nu} < 0; \text{and} \\
  (13) & \quad \{p + 1,i\} \in E.
\end{align*}
\]

Moreover, it follows from \( a_{p+1,p}^{\nu} < 0 \) (since \( x_{p+1} \) is now the driving variable), and from point 3 of the inductive assumption for \( s = p + 1 \)

\[
a_{p+1,i}^{\nu} = \frac{1}{2w_i} \tilde{d}_{i}(p + 1) = 0,
\]

because of (13). It follows from (8) with \( s = t = i \) and (14) that

\[
a_{i,i}^{\nu+1} = a_{i,i}^{\nu} + 2w_i a_{p+1,i}^{\nu} a_{i,p}^{\nu} = a_{i,i}^{\nu} = \frac{1}{2w_i},
\]

where the last equality follows from (12) and point 2 of the inductive assumption. Then point 2 is satisfied also for \( \nu + 1 \).

**Point 3** Again let us assume that \( a_{p+1,p}^{\nu+1} < 0 \). As in the proof for the previous point, we obtain (12) and (14). Now for any \( t \not\in K \cup \{s,p,p+1\} \)

\[
a_{s,i}^{\nu+1} = a_{s,i}^{\nu} + 2w_i a_{p+1,i}^{\nu} a_{s,p}^{\nu} = a_{s,i}^{\nu}
\]

where the last equality follows from (14). But since (12) holds, point 3 of the inductive assumption implies that

\[
a_{s,t}^{\nu} = \frac{1}{2w_i} \tilde{d}_{i}(s),
\]

and point 3 also holds for \( \nu + 1 \). \( \Box \)

**Proposition 2.3.** At iteration \( \nu \) of the PBH algorithm, let \( x_p \) be the driving variable and \( K = \{i \in V : x_i \text{ is basic}\} = \{1, \ldots, p - 1\} \). Then, the following statements are true:

1. For any \( j \in K \)

\[
a_j^{\nu+1} = \frac{w_i}{w_p}.
\]

2. If \( a_{i,p}^{\nu} < 0, t \neq 1 \), then

\[
\forall j \in K : \quad a_{j,t}^{\nu+1} = 0.
\]

3. If \( a_{i,p}^{\nu} < 0, t \neq 1 \), then

\[
a_{1,t}^{\nu} = a_{1,t}^{1} = -1.
\]
4.

\[ a_{i,p}^\nu = \frac{1}{2w_p} [1 - 2W(K \cup \{p\})]. \]

Proof. The proof is again by induction. We first prove that the result is true at iteration \( \nu = 1 \) and then we assume that it is true at iteration \( \nu \geq 1 \) and we prove it is true at iteration \( \nu + 1 \).

\( \nu = 1 \)

**Points 1 and 2** Points 1 and 2 are trivially satisfied because at iteration \( \nu = 1 \) it holds that \( K = \emptyset \).

**Point 3** Since, by assumption, the first driving variable is \( x_1 \) (i.e., \( p = 1 \)) and \( t \neq 1 \), then \( a_{i,1}^1 = q_1 - q_{11} = \frac{1}{2w_1} [d_{11}(t) - 1] \). If \( a_{i,1}^1 < 0 \), then \( (t,1) \in E \). Thus

\[ a_{i,1}^1 = q_{11} - 1 = q_{11} - 1 = -1 \]

and point 3 is satisfied.

**Point 4** Finally, since \( K = \emptyset \),

\[ a_{i,1}^1 = q_{11} - 1 = \frac{1}{2w_1} [1 - 2W(K \cup \{1\})]. \]

From \( \nu \) to \( \nu + 1 \)

Again we assume that the new driving variable at iteration \( \nu + 1 \) is \( x_{p+1} \).

**Point 1** For any \( j \in K \) it holds that

\[ a_{j+1,p+1}^{\nu+1} = a_{j+1,p+1}^\nu + 2w_p a_{p+1,j+1,p}^\nu a_{j+1,p}^\nu. \]

Now since \( x_{p+1} \) is the driving variable, we have \( a_{p+1,j}^\nu < 0 \) and from point 2 of Proposition 2.2 with \( i = p + 1 \notin K \cup \{p\} \), we have \( a_{p+1,j}^\nu = \frac{1}{2w_{p+1}} \). Moreover, point 2 of the inductive assumption implies \( a_{j+1,p}^\nu = 0 \). Therefore

\[ a_{j+1,p+1}^{\nu+1} = a_{j+1,p+1}^\nu + \frac{w_p}{w_{p+1}} a_{j+1,p}^\nu = \frac{w_p}{w_{p+1}} a_{j+1,p}^\nu. \]

In view of point 1 of the inductive assumption, it holds that \( a_{j+1,p}^\nu = \frac{w_j}{w_p} \). Thus, point 1 is satisfied also for \( \nu + 1 \) for any \( j \in K \). We still need to prove it for \( j = p \).

But in this case, (9) for \( t = p + 1 \) implies

\[ a_{p+1,p+1}^{\nu+1} = 2w_p a_{p+1,p+1}^\nu = \frac{2w_p}{w_{p+1}} a_{p+1,p+1}^\nu, \]

and point 1 is true also for \( j = p \).

**Point 2** First we notice that in the proof of point 1 we have just shown that for \( j \in K \cup \{p\} = \{1, \ldots, p\} \) it holds that \( a_{j+1,p+1}^{\nu+1} = \frac{w_j}{w_{p+1}} > 0 \). This also means that for \( t = j + 1 \), the relation \( a_{j+1,p+1}^{\nu+1} < 0 \) is impossible for \( t \in K \cup \{p\} \subset \{2, \ldots, p + 1\} \) (unless \( t = 1 \), which is excluded). Hence \( a_{j+1,p+1}^{\nu+1} < 0 \) and \( t \neq 1 \) implies \( t \notin K \cup \{p\} \). Now, we
recall that in the proof of point 2 in Proposition 2.2 we showed that (putting $t = i$)
\[ a_{i,p+1}^{v+1} < 0 \text{ implies } a_{i+1,t}^{v+1} = 0 \text{ due to (14)} \] and \[ a_{i,p}^{v+1} < 0 \text{ due to (12)} \]. Next we notice that for any $j \in K$ it follows from (8) that
\[ a_{j+1,t}^{v+1} = a_{j+1,t}^{v} + 2w_p a_{j+1,p}^{v} a_{j+1,t}^{v} = a_{j+1,t}^{v} . \]
But since $a_{i,p}^{v+1} < 0$, point 2 of the inductive assumption implies that $a_{j+1,t}^{v+1} = 0$ and point 2 is satisfied also for $\nu + 1$ for any $j \in K$. We still need to prove that it is true for $j = p$. But in such case it follows from (9) that
\[ a_{p+1,t}^{\nu+1} = 2w_p a_{p+1,t}^{\nu+1} = 0, \]
and point 2 is satisfied also for $j = p$.

**Point 3** For any $t \neq 1$ such that $a_{i, p+1}^{v+1} < 0$ we have $a_{p+1,t}^{v+1} = 0$ and thus it holds that
\[ a_{i,t}^{v+1} = a_{i,t}^{v} + 2w_p a_{p+1,t}^{v} a_{i,t}^{v} = a_{i,t}^{v} , \]
and since $a_{i,p}^{v} < 0$ (refer, again, to the proof of Proposition 2.2), point 3 of the inductive assumption implies $a_{i,t}^{v} = -1$, so that point 3 is satisfied also for $\nu + 1$.

**Point 4** Note that $a_{i, p+1}^{v+1} < 0$ implies $a_{i, p+1}^{v} = -1$ in view of point 3 of the inductive assumption, while $a_{p+1, p+1}^{v} = \frac{1}{w_{p+1}}$ (see Proposition 2.2). Then,
\[ a_{i, p+1}^{v+1} = a_{i,p+1}^{v} + 2w_p a_{p+1,p+1}^{v} a_{i,p}^{v} = -1 + \frac{w_p}{w_{p+1}} a_{i,p}^{v} . \]
But by point 4 of the inductive assumption, $a_{i,p}^{v} = \frac{1}{2w_p} [1 - 2W(K \cup \{p\})]$. Thus
\[ a_{i,p+1}^{v+1} = -1 + \frac{1}{2w_{p+1}} [1 - 2W(K \cup \{p\})] = \frac{1}{2w_{p+1}} [1 - 2W(K \cup \{p,p+1\})] , \]
and point 4 is satisfied also for $\nu + 1$.

### 2.4. Algorithmic consequences.
Now we are ready to summarize the consequences of the results in Propositions 2.2 and 2.3.

1. As conjectured in [11], once a $x_i$ variable with $i \in V$ enters the basis, it never exits. This follows from point 1 of Proposition 2.3 where each entry $a_{j+1,p}^{v}$ in the column of the driving variable and related to a variable $x_j$ in the basis is equal to $\frac{u}{w}$ so that $x_j$ cannot exit the basis.

2. The candidates to become the new driving variable, i.e. with $a_{i,p}^{v} < 0$ are only the variables related to vertices $i$ which form a clique with the vertices in $K \cup \{p\}$. This follows from point 1 of Proposition 2.2 from which $a_{i,p}^{v} < 0$ if and only if $[K \cup \{p\}]_i = \emptyset$ which exactly means that $i$ is connected to every vertex in $K \cup \{p\}$.

3. The rule to choose the next driving variable in [11] is the following greedy rule. Let $H$ the set of candidates to become the new driving variable, i.e.,
\[ H = \{ j : a_{j,p}^{v+1} < 0 \} = \bigcap_{i \in K \cup \{p\}} N_i . \]
For each $j \in H$ let
\[ U(j) = \{ s \in H \setminus \{j\} : \{s,j\} \in E \} = H \cap N_j . \]
Then, the new driving variable is selected in the set
\[ \arg \max \{|U(j)| : j \in H \}, \]
and, more precisely, as the variable with the lowest index in this set.

Note that the algorithm always stops if and only if \( H = \emptyset \) or, equivalently, when \( K \) is a maximal clique. This proves the conjecture stated in [11] that saddle points detected by the algorithm are always maximal cliques, and thus local solutions, even if program (1) has stationary points which are not local minimizers.

3. Experience with new heuristic approaches.

3.1. Description of the heuristics. In the previous section we proved that the continuous approach PBH for the solution of the MWCP is equivalent to a combinatorial heuristic. In the unweighted case, to which we shall now restrict our attention, such a heuristic turns out to belong to the class of greedy heuristics denoted in [5] by \( SM^i \quad (i = 1, 2, \ldots) \), which are described in what follows.

**Greedy heuristic \( SM^i \)**

**Step 1** Given a graph \( G = (V, E) \), let \( Q \) be the set of all cliques of graph \( G \) with cardinality \( i \). Set \( K = \emptyset \), \( K^* = \emptyset \) and \( \text{max} = 0 \).

**Step 2** If \( Q \neq \emptyset \), select a clique \( H \in Q \) and update \( Q = Q \setminus \{H\} \). Put \( K = H \) and go to Step 3; otherwise, return \( \text{max} \) and \( K^* \).

**Step 3** If \( K \) is a maximal clique, then update \( K = K \cup \{K\} \) and go to Step 4; otherwise go to Step 5.

**Step 4** If \( |K| > \text{max} \), set \( \text{max} = |K| \) and \( K^* = K \). Go back to Step 2.

**Step 5** Select a vertex \( \ell \in \arg \max_{j \in C_0(K)} |C_0(K) \cap N_j| \)

at random, where

\[ C_0(K) = \{j \in V : (j, k) \in E \quad \forall \ k \in K\} = \bigcap_{k \in K} N_k \]

is the set of all vertices in \( G \) adjacent to each vertex in the current clique \( K \).

**Step 6** Set \( K = K \cup \{\ell\} \) and go back to Step 3.

We notice that the combinatorial version of PBH in the unweighted case is equivalent to \( SM^1 \), except for the minimal difference of the random selection of \( \ell \) in (15)—in PBH \( \ell \) is selected as the node with the lowest index in \( C_0(K) \). Of course, although the heuristic \( SM^1 \) frequently returns results of good quality, the heuristic \( SM^2 \) is often superior to \( SM^1 \) (see Table 2). On the other hand, as expected, the computation times for \( SM^2 \) are much larger than those for \( SM^1 \) (see Table 3). The aim of this section is that of proposing modifications of the heuristic \( SM^1 \) in such a way that, at the price of larger computation times, we will also be able to obtain better results. Basically, we would like to modify the heuristic \( SM^1 \) in such a way that the modification is closer to \( SM^2 \) than to \( SM^1 \) from the point of view of the quality of the results, but is much closer to \( SM^1 \) than to \( SM^2 \) from the point of view of the computation times.

The first modification proposed here is to allow vertex swaps while we build a maximal clique. The heuristics \( SM^i \) start with a clique of cardinality \( i \) and add a new vertex at each iteration until a maximal clique is reached. Some other approaches do the same but do not stop once a maximal clique is reached, allowing for the
removal of vertices when this situation occurs (see e.g. the plateau search in [2]). The modification proposed here does not always add a new vertex to the current clique but also allows to swap vertices. If $K$ is the current clique, let $C_0(K)$ be defined as in (16) and let

$$C_1(K) = \{j \in V \setminus K : |N_j \cap K| = |K| - 1\},$$

i.e. $C_1(K)$ is the set of vertices outside $K$ which are connected to all vertices in $K$ but exactly one. If we select a vertex $\ell \in C_0(K)$, then we can add it to $K$ and update $K$ as follows

$$K = K \cup \{\ell\}.$$

This is exactly what heuristics $SM^i$ do with the selection (15). But if we select a vertex $\ell \in C_1(K)$ we can not simply add it to $K$; if we still want to have a clique we need to swap vertex $\ell$ with the unique vertex

$$k_\ell \in K \quad \text{such that} \quad (\ell, k_\ell) \notin E,$$

i.e. the new clique will be

$$K = K \cup \{\ell\} \setminus \{k_\ell\}.$$

Therefore, if we select a vertex in $C_1(K)$ we change the current clique but without increasing the cardinality, i.e. the method is not strictly monotone. In the modification proposed here we decide whether to increase the cardinality, by selecting a vertex in $C_0(K)$, or to swap vertices, by selecting a vertex in $C_1(K)$. Note that $C_0(K) = \emptyset$ if $K$ is a maximal clique. Similarly, some authors call a maximal clique $K$ strictly maximal, if $C_1(K) = \emptyset$, i.e., if every swap destroys the clique property.

Now, selection of a vertex in $C_0(K) \cup C_1(K)$ is done according to a rule similar to (15) but with the computation of the maximum extended also to vertices in $C_1(K)$, i.e.

$$\ell \in \arg\max_{j \in C_0(K) \cup C_1(K)} |C_0(K) \cap N_j|.$$

We underline that in some cases we force the heuristic to select the next vertex according to (15), namely:

- during a fixed number ($\text{START}_\text{SWAP}$) of initial iterations;
- when the number of swaps performed is a fixed number $T$ times the cardinality of the current clique $|K|$;
- when the vertex selected in $C_1(K)$ is the same as the last one removed by the last swap. Without this condition, which can be interpreted as a very limited application of taboo rules, it has been observed that the heuristic is much less efficient (worse results and larger computation times), apparently because it happens that the same two vertices are swapped in and out until the maximum allowed number of swaps is reached and then the heuristic behaves exactly as the standard greedy one.

The heuristic with the proposed modification, denoted with $SM^1_\text{SWAP}$ is described in what follows.

**Heuristic $SM^1_\text{SWAP}$**

**Step 1** Given a graph $G = (V, E)$, let $W = V$, $K^* = \emptyset$ and $max = 0$. Select a value for the parameters $\text{START}_\text{SWAP}$ and $T$. 
Step 2 If \( W = \emptyset \) return \( K^* \) and \( \max \). Otherwise select a vertex \( h \in W \), set \( W = W \setminus \{ h \} \) and set \( K = \{ h \} \). Set \( n_{\text{swaps}} = 0 \) and \( \text{last swap} = \emptyset \).

Step 3 If \( K \) is a maximal clique, then set \( K = K \cup \{ h \} \) and go to Step 4; otherwise go to Step 5.

Step 4 If \( |K| > \max \), set \( \max = |K| \) and \( K^* = K \). Go back to Step 2.

Step 5 If \( n_{\text{swaps}} \leq T|K| \) and \( |K| \geq \text{START SWAP} \), then go to Step 6, otherwise go to Step 7.

Step 6 Select randomly \( \ell \in V \) according to (18). If \( \ell = \text{last swap} \), then go to Step 7, otherwise go to Step 8.

Step 8 If \( \ell \in C_1(K) \), then update

\[
\begin{align*}
n_{\text{swaps}} &= n_{\text{swaps}} + 1 \\
\text{last swap} &= k_\ell
\end{align*}
\]

where \( k_\ell \) is defined in (17), and

\[
K = K \cup \{ \ell \} \setminus \{ k_\ell \}.
\]

Otherwise set

\[
K = K \cup \{ \ell \}.
\]

Go back to Step 3.

3.2. Benchmark results and comparisons. We consider a selection of DIMACS benchmark graphs [10][1] (some benchmarks, such as the c-fat and johnson graph classes and most of the hamming graphs, have been omitted because their solutions were easily detected by all the tested heuristics). In Table 2 we report the average, worst and best results for the heuristics \( SM^1 \), \( SM^1\_SWAP \) and \( SM^2 \) over 10 random runs, while in Table 3 we report the the average computation times. For the heuristic \( SM^1\_SWAP \) we employed the following parameter values: \( \text{START SWAP} = 5 \), \( T = 2 \).

We notice that, as required, the results for \( SM^1\_SWAP \) are often competitive with those of \( SM^2 \), while the computation times are much closer to those of \( SM^1 \). If we compare the results of \( SM^1 \) with those of \( SM^1\_SWAP \), we notice that \( SM^1\_SWAP \) often outperforms \( SM^1 \). With the only exception of the \( p\hat{\text{hat}1500-1} \) graph, we notice that \( SM^1\_SWAP \) largely outperforms \( SM^1 \) over most of the cliques in the \( p\hat{\text{hat}} \) class. The same is true for many cliques in the \( san \) class and for the \( \text{hamming10-4} \) graph. It is remarkable that, in spite of the much lower computational times, the heuristic \( SM^1\_SWAP \) turns out to be superior even to \( SM^2 \) over three graphs, namely \( p\hat{\text{hat}1000-3} \), \( p\hat{\text{hat}1500-3} \) and \( \text{hamming10-4} \). We remark that the best known clique size (68) reported for the \( p\hat{\text{hat}1000-3} \) graph has, to the authors’ knowledge, only been very recently detected in [6].

We also note that changing the parameter \( \text{START SWAP} \) from 5 to 2 turned out to be profitable in some cases. In particular the average result for \( keller6 \) has been 55.7 with a minimum of 55 and a maximum of 57. Finally, note that on \( keller6 \) even a single run of \( SM^2 \) did not terminate after a whole day and hence we stopped it prematurely (this has been indicated with a hyphen in Tables 2 and 3).

Now we compare our results to those reported in four other papers, some of them classics, some published recently, and one still unpublished. By mere coincidence,
Table 2
Average (worst, best) results over 10 random tests for SM₁, SM₁_Swap and SM₂. When only a single value is reported, all the random tests returned that single value.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Best (known) value</th>
<th>SM₁</th>
<th>SM₁_Swap</th>
<th>SM₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>pHat300-1</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>pHat300-2</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>pHat300-3</td>
<td>36</td>
<td>35</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>pHat500-1</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>pHat500-2</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>pHat500-3</td>
<td>≥ 50</td>
<td>49</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>pHat700-1</td>
<td>11</td>
<td>10.2 (9,11)</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>pHat700-2</td>
<td>44</td>
<td>43.8 (43,44)</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>pHat700-3</td>
<td>≥ 62</td>
<td>62</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>pHat1000-1</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>pHat1000-2</td>
<td>46</td>
<td>46</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>pHat1000-3</td>
<td>≥ 68</td>
<td>64</td>
<td>68</td>
<td>66.8 (66,67)</td>
</tr>
<tr>
<td>pHat1500-1</td>
<td>12</td>
<td>12</td>
<td>11.4 (11,12)</td>
<td>12</td>
</tr>
<tr>
<td>pHat1500-2</td>
<td>≥ 65</td>
<td>64</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>pHat1500-3</td>
<td>≥ 94</td>
<td>94</td>
<td>94</td>
<td>93</td>
</tr>
<tr>
<td>MANN27</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>126</td>
</tr>
<tr>
<td>MANN45</td>
<td>345</td>
<td>343.8 (343,344)</td>
<td>343.8 (343,344)</td>
<td>344</td>
</tr>
<tr>
<td>keller4</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>keller5</td>
<td>27</td>
<td>27</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>keller6</td>
<td>≥ 59</td>
<td>54.4 (54,55)</td>
<td>55.4 (55,56)</td>
<td>—</td>
</tr>
<tr>
<td>brock200-1</td>
<td>21</td>
<td>20.6 (20,21)</td>
<td>20.9 (20,21)</td>
<td>21</td>
</tr>
<tr>
<td>brock200-2</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>brock200-3</td>
<td>15</td>
<td>14</td>
<td>13.8 (13,14)</td>
<td>15</td>
</tr>
<tr>
<td>brock200-4</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>brock400-1</td>
<td>27</td>
<td>23.8 (23,24)</td>
<td>24.2 (24,25)</td>
<td>25</td>
</tr>
<tr>
<td>brock400-2</td>
<td>29</td>
<td>24</td>
<td>24.5 (24,25)</td>
<td>29</td>
</tr>
<tr>
<td>brock400-3</td>
<td>31</td>
<td>24</td>
<td>24.9 (24,25)</td>
<td>31</td>
</tr>
<tr>
<td>brock400-4</td>
<td>33</td>
<td>24</td>
<td>24.7 (24,25)</td>
<td>33</td>
</tr>
<tr>
<td>brock800-1</td>
<td>23</td>
<td>21</td>
<td>20.2 (20,21)</td>
<td>21</td>
</tr>
<tr>
<td>brock800-2</td>
<td>24</td>
<td>20</td>
<td>20.7 (20,21)</td>
<td>21</td>
</tr>
<tr>
<td>brock800-3</td>
<td>≥ 25</td>
<td>20.8 (20,21)</td>
<td>20.9 (20,22)</td>
<td>22</td>
</tr>
<tr>
<td>brock800-4</td>
<td>≥ 26</td>
<td>20.2 (20,21)</td>
<td>20.2 (20,21)</td>
<td>21</td>
</tr>
<tr>
<td>san_200_0.7_1</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>san_200_0.7_2</td>
<td>18</td>
<td>17.4 (17,18)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>san_200_0.9_1</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>san_200_0.9_2</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>san_200_0.9_3</td>
<td>44</td>
<td>41.4 (40,44)</td>
<td>44</td>
<td>44</td>
</tr>
<tr>
<td>san_400_0.5_1</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>san_400_0.7_1</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>san_400_0.7_2</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>san_400_0.7_3</td>
<td>22</td>
<td>17.8 (17,19)</td>
<td>19.9 (17,22)</td>
<td>22</td>
</tr>
<tr>
<td>san_400_0.9_1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>sanr_200_0.7</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>sanr_200_0.9</td>
<td>42</td>
<td>41</td>
<td>41.9 (41,42)</td>
<td>42</td>
</tr>
<tr>
<td>sanr_400_0.5</td>
<td>13</td>
<td>12.4 (12,13)</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>sanr_400_0.7</td>
<td>≥ 21</td>
<td>20</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>san1000</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>hamming10-4</td>
<td>≥ 40</td>
<td>36</td>
<td>40</td>
<td>36</td>
</tr>
</tbody>
</table>
Table 3
Average computation times (in seconds) over 10 random tests for $SM^1$, $SM^1_{\text{SWAP}}$ and $SM^2$.

<table>
<thead>
<tr>
<th>Graph</th>
<th>$SM^1$</th>
<th>$SM^1_{\text{SWAP}}$</th>
<th>$SM^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_hat300-1</td>
<td>0.03</td>
<td>0.11</td>
<td>0.73</td>
</tr>
<tr>
<td>p_hat300-2</td>
<td>0.15</td>
<td>0.66</td>
<td>7.04</td>
</tr>
<tr>
<td>p_hat300-3</td>
<td>0.31</td>
<td>1.19</td>
<td>22.75</td>
</tr>
<tr>
<td>p_hat500-1</td>
<td>0.12</td>
<td>0.46</td>
<td>4.26</td>
</tr>
<tr>
<td>p_hat500-2</td>
<td>0.64</td>
<td>3.04</td>
<td>50.96</td>
</tr>
<tr>
<td>p_hat700-1</td>
<td>1.37</td>
<td>5.70</td>
<td>166.72</td>
</tr>
<tr>
<td>p_hat700-2</td>
<td>0.25</td>
<td>1.03</td>
<td>12.59</td>
</tr>
<tr>
<td>p_hat700-3</td>
<td>1.86</td>
<td>7.84</td>
<td>181.86</td>
</tr>
<tr>
<td>p_hat1000-1</td>
<td>3.87</td>
<td>13.53</td>
<td>612.05</td>
</tr>
<tr>
<td>p_hat1000-2</td>
<td>0.61</td>
<td>2.39</td>
<td>39.52</td>
</tr>
<tr>
<td>p_hat1000-3</td>
<td>4.26</td>
<td>18.93</td>
<td>605.61</td>
</tr>
<tr>
<td>MANNn27</td>
<td>10.34</td>
<td>36.33</td>
<td>2324.86</td>
</tr>
<tr>
<td>MANNn45</td>
<td>102.16</td>
<td>659.05</td>
<td>46273.89</td>
</tr>
<tr>
<td>keller4</td>
<td>0.02</td>
<td>0.10</td>
<td>0.88</td>
</tr>
<tr>
<td>keller5</td>
<td>1.74</td>
<td>8.81</td>
<td>380.68</td>
</tr>
<tr>
<td>keller6</td>
<td>373.8</td>
<td>1651.2</td>
<td>—</td>
</tr>
<tr>
<td>brock200-1</td>
<td>0.07</td>
<td>0.24</td>
<td>3.63</td>
</tr>
<tr>
<td>brock200-2</td>
<td>0.03</td>
<td>0.09</td>
<td>0.80</td>
</tr>
<tr>
<td>brock200-3</td>
<td>0.05</td>
<td>0.14</td>
<td>1.51</td>
</tr>
<tr>
<td>brock200-4</td>
<td>0.05</td>
<td>0.16</td>
<td>2.04</td>
</tr>
<tr>
<td>brock400-1</td>
<td>0.41</td>
<td>1.25</td>
<td>37.57</td>
</tr>
<tr>
<td>brock400-2</td>
<td>0.45</td>
<td>1.25</td>
<td>37.69</td>
</tr>
<tr>
<td>brock400-3</td>
<td>0.43</td>
<td>1.25</td>
<td>39.08</td>
</tr>
<tr>
<td>brock400-4</td>
<td>0.43</td>
<td>1.28</td>
<td>37.87</td>
</tr>
<tr>
<td>brock800-1</td>
<td>1.71</td>
<td>4.90</td>
<td>217.42</td>
</tr>
<tr>
<td>brock800-2</td>
<td>1.73</td>
<td>5.02</td>
<td>219.61</td>
</tr>
<tr>
<td>brock800-3</td>
<td>1.69</td>
<td>4.88</td>
<td>221.50</td>
</tr>
<tr>
<td>brock800-4</td>
<td>1.73</td>
<td>4.76</td>
<td>213.87</td>
</tr>
<tr>
<td>sanr200-0.7_1</td>
<td>0.07</td>
<td>0.43</td>
<td>3.28</td>
</tr>
<tr>
<td>sanr200-0.7_2</td>
<td>0.06</td>
<td>0.35</td>
<td>2.66</td>
</tr>
<tr>
<td>sanr200-0.9_1</td>
<td>0.23</td>
<td>1.29</td>
<td>16.56</td>
</tr>
<tr>
<td>sanr200-0.9_2</td>
<td>0.20</td>
<td>0.97</td>
<td>12.92</td>
</tr>
<tr>
<td>sanr200-0.9_3</td>
<td>0.15</td>
<td>0.58</td>
<td>10.63</td>
</tr>
<tr>
<td>sanr200-0.9_4</td>
<td>0.12</td>
<td>0.55</td>
<td>7.30</td>
</tr>
<tr>
<td>sanr200-0.7_1</td>
<td>0.44</td>
<td>2.54</td>
<td>39.49</td>
</tr>
<tr>
<td>sanr200-0.7_2</td>
<td>0.36</td>
<td>2.11</td>
<td>30.53</td>
</tr>
<tr>
<td>sanr200-0.7_3</td>
<td>0.29</td>
<td>1.35</td>
<td>25.03</td>
</tr>
<tr>
<td>sanr200-0.9_1</td>
<td>1.30</td>
<td>6.45</td>
<td>180.41</td>
</tr>
<tr>
<td>sanr200-0.7</td>
<td>0.06</td>
<td>0.19</td>
<td>2.59</td>
</tr>
<tr>
<td>sanr200-0.9</td>
<td>0.18</td>
<td>0.61</td>
<td>11.68</td>
</tr>
<tr>
<td>sanr400-0.5_1</td>
<td>0.14</td>
<td>0.49</td>
<td>8.15</td>
</tr>
<tr>
<td>sanr400-0.7_1</td>
<td>0.33</td>
<td>1.02</td>
<td>27.10</td>
</tr>
<tr>
<td>san1000</td>
<td>1.35</td>
<td>7.85</td>
<td>192.38</td>
</tr>
<tr>
<td>hamming10-4</td>
<td>4.83</td>
<td>28.19</td>
<td>1606.12</td>
</tr>
</tbody>
</table>
all first authors' names start with a "B" and their lexicographic order coincides with the historic one. The classic contribution by Balas and Niethaus [1] treats 20 cases occurring in the above list (e.g., all graphs in the san class are missing), and is considered to be one of the dominating results of the DIMACS challenge [10]. Their results are still, ten years ago, competitive, however, are dominated by the reactive local search procedure of Battiti and Protasi [2] who report on the same subset of 20 instances. The latter specify runtimes to achieve the best observed solution, with averages and standard deviation of the same order of magnitude (sometimes with larger deviations), which indicates a highly variable runtime behaviour that is prone to relatively large outliers. Since we report, in Table 3, the overall average runtime, it is not surprising that the times reported in [2] are comparatively small. Still, there is an exception, the brock instances, where the times in [2] consistently exceed even those in $SM^2$, save two 800-vertex cases where they are comparable to the $SM^1\_SWAP$ times. In a paper to appear [6], Burer and co-workers provide results for all cases covered here, which are slightly dominated by those in Table 1. However, comparing their runtimes with those of Table 2, one sees that their procedure is considerably slower even, with very few exceptions, compared to $SM^2$. Finally, the unpublished results by Busygin [7] are also dominated by those presented here, with exception of his really impressive achievements on the large brock instances, which also beat all other approaches discussed here. Runtimes reported in [7] are roughly within the same order of magnitude as those in Table 2, only for the brock, some phat and the hamminig instances his procedure is slower by a factor ranging between four and ten. Of course, comparing runtimes is always problematic due to differences in implementation, architecture and hardware platforms. Our experiments have been performed on a 1 GHz Pentium III under Linux 2.4 with 240 MB RAM.

3.3. Diversification. The only benchmarks over which we did not achieve very satisfying results were those belonging to the brock class. However, the less satisfying results over the brock graphs do not come as a surprise. The authors of [5], who proposed the brock class, explicitly say that it was their intention to create difficult problems for the class of greedy heuristics $SM^1$. We notice that $SM^1$ is unable to detect all except one of the maximum cliques for the twelve brock graphs. We can also notice that $SM^2$ is unable to detect the maximum clique for the brock400_1 graph and for all the brock800 graphs. Unfortunately the introduction of swaps does not significantly improve the results of $SM^1$. Therefore, we would like to propose a further modification in order to be able to reach results at least comparable to that of $SM^2$. While we run a heuristic such as $SM^1\_SWAP$ we collect in the set $K$, updated in Step 3, all the maximal cliques generated. Some information, which for the moment we only denote by $Info$ without specifying it in more detail, can be extracted from $K$. Once we have extracted such information, we can use it to run the less time consuming heuristic $SM^1$ but by changing rule (15) for the selection of a new vertex $\ell$. Namely, the rule to select the next vertex $\ell$ among vertices $j \in C_0(K)$ could be some function of $C_0(K)$, $N_j$ but also of the information $Info$. Therefore we could modify heuristic $SM^1$ by modifying Step 5 as follows:

**Step 5** Randomly select a vertex $\ell \in C_0(K)$ according to the following rule:

$$\ell \in \arg \max_{j \in C_0(K)} \nu(C_0(K), N_j, Info).$$

Of course we still need to specify how to extract the information $Info$ from $K$ and how to use it in order to define the function $\nu$. The kind of information we decide
to extract depends on how we intend to use it. Here we propose to use it in order to diversify the search for cliques. In order to accomplish such a diversification of the search we extract from $\mathcal{K}$ the following information

$$In_j = \{S_j : j \in V\},$$

where, for any $j \in V$

$$S_j = \{i \in N_j : \{i, j\} \notin K \ \forall K \in \mathcal{K}\},$$

e.i. $S_j$ is the set of vertices adjacent to vertex $j$ which never appear together with $j$ in any of the maximal cliques generated by heuristic $SM^1_{\text{SWAP}}$. The above information is used as follows to define the function $\nu$ employed in Step 5'

$$\nu(C_0(K), N_j, In_j) = |C_0(K) \cap N_j| + |C_0(K) \cap S_j|$$

The first term in the sum is the usual one favouring, according to the greedy principle, vertices with a lot of neighbors in $C_0(K)$. But the second term favours a vertex $j$ if many of its neighbors in $C_0(K)$ have never been together with $j$ in the cliques generated by heuristic $SM^1_{\text{SWAP}}$ (i.e. vertices belonging to $S_j$). Therefore, function (21) is a compromise between the standard greedy rule and a term which favours the generation of cliques different from those previously generated by heuristic $SM^1_{\text{SWAP}}$. This is basically equivalent to changing the weights of the vertices in the graph. While in $SM^1$ each vertex has weight 1 and $|C_0(K) \cap N_j|$ is equivalent to the sum of the weights of the vertices in $C_0(K)$ which are adjacent to vertex $j$, in the proposed modification a vertex $i \in C_0(K)$ has weight 1 if $i \not\in S_j$ and weight 2 if $i \in S_j$. The modification of weights after each restart of a greedy algorithm has been already explored in the adaptive algorithms presented in [9]. However, we underline that while in [9] the weight of a vertex is the same with respect to any other vertex, in the modification proposed in (21) each vertex has not a fixed weight but its weight is relative to other vertices. The same vertex $i$ may have weight 1 with respect to a vertex $j_1$ (if $i \not\in S_{j_1}$) but it may have weight 2 with respect to a different vertex $j_2$ (if $i \in S_{j_2}$).

We remark that running again $SM^1$ with the modified Step 5' after having run heuristic $SM^1_{\text{SWAP}}$ only slightly increases the computation times with respect to those of $SM^1_{\text{SWAP}}$. In Table 4 we report the results and the computation times obtained by running heuristic $SM^1_{\text{SWAP}}$ followed by heuristic $SM^1$ with the modified Step 5' over the brock graphs. We notice that with a slight increase in the computation times with respect to $SM^1_{\text{SWAP}}$ we are now able to detect the maximum clique for many brock graphs. The exceptions are the brock400,1 graph and the brock800 graphs, but on these graphs even the much more time consuming heuristic $SM^2$ fails. Again, changing the parameter $START_{\text{SWAP}}$ from 5 to 2 turned out to be profitable in some cases. In particular for brock400,4 the optimal value 33 has been detected in all the random runs and for brock300,4 the average increased to 16.3.

After presenting the successes of such a modification we also need to underline its failures. On the other graphs for which $SM^1_{\text{SWAP}}$ was not always able to return the maximum clique, the modification proposed above did not lead to any improvement (with the only exception of the MANNA45 graph for which an improvement from an average of 343.8 to 344 was accomplished). This is probably due to the fact that the information extracted as in (20) and its use in (21) are not always as appropriate
as they are for the brock graphs. However, the successes obtained with such graphs justify the hope that ways other than (20) of extracting information and inserting it into the definition of a function $\nu$, could lead to further good results. We believe that gaining insight into the reasons of the successes (and the failures) of the information extracted and the ways it is employed could be an interesting issue for future research.

REFERENCES


