CAHIER DU LAMSADE

L'Analyse et Modélisation de Systèmes pour l'Aide à la Décision.
Contents

I  A polynomial algorithm for the maximum independent set problem on a class of graphs including König-Egerváry graphs 2
Résultats exacts et approchés sur les problèmes du stable maximum et de la couverture minimum de sommets - graphes à grand nombre de stabilité

Dans la première partie de cet article, nous présentons un algorithme exact polynomial pour...
Exact and approximation results on maximum independent set and minimum vertex covering - graphs with great stability number

Abstract

In the first part of this paper, we present an exact polynomial time algorithm for maximum independent set problem in a class of graphs including König-Egervary graphs. The class of König-Egervary graphs is the class of graphs with $\alpha(G) = n - m$, where given a graph $G$ of order $n$, we denote by $\alpha$ its stability number and by $m$ the cardinality of a maximum matching of $G$.

Next, we prove that the existence of a polynomial time $\rho$-approximation algorithm (where $\rho < 1$ is a fixed constant), for a class of independent set problems, leads to a polynomial time approximation algorithm with approximation ratio strictly smaller than 2 for vertex covering, while the non-existence of such an algorithm induces a lower bound on the ratio of every polynomial time approximation algorithm for vertex covering. We also prove a similar result for a (maximisation) convex programming problem including quadratic programming as subproblem.

Finally, we show that the natural greedy algorithm for maximum independent set problem
Introduction

Consider a graph $G = (V, E)$ of order $n$. An independent set is a subset $S \subseteq V$ such that no two vertices in $S$ are linked by an edge in $G$; for $\kappa > 1$, let us denote by $S_\kappa$ the following problem: "given a graph $G$ admitting a maximum independent set of cardinality greater than or equal to $n/\kappa$, find a maximum independent set of $G$.

A vertex covering is a subset $C \subseteq V$ such that for
Part I

A polynomial algorithm for the maximum independent set problem on a class of graphs including König-Egerváry graphs

In [4], Bourjolly et al. produce results similar to the ones of this section. Here, we give simpler algorithms and proofs and, moreover, we bring to the fore some properties of the graphs of the considered class (as, for example, the value of the "discrete duality gap") not exhaustively considered in [4], properties that permit us to obtain some further exact or approximate polynomial results for S and VC. On the other hand, for the completeness of the paper and since we use the results of this section as a subcase result of theorem 5 of section 6, we give all of our proofs in details. We also notice that, in [5], Deming gives an exact polynomial time algorithm for S (and consequently for VC) for the class of KE graphs.

1 From independent set to matching

A general instance\(^1\) of S defined by a graph \(G = (V, E)\) can be written as a \((0,1)\) linear problem as follows:

\[
S = \begin{cases} 
\max \vec{I}_n \cdot \vec{x} \\
A \cdot \vec{x} \leq \vec{l}[E] \\
\vec{x} \in \{0, 1\}^n
\end{cases}
\]

where \(A\) is the edge-vertex incidence matrix of \(G\) and \(\vec{l}_D (\vec{0}_D)\) is the one-column vector of \(R^D\) \((D \in \mathbb{N})\), all of its coordinates being equal to 1 (0).

Let us denote by SR the following relaxed version of S:

\[
SR = \begin{cases} 
\max \vec{I}_n \cdot \vec{x} \\
A \cdot \vec{x} \leq \vec{l}[E] \\
\vec{x} \geq \vec{0}_n
\end{cases}
\]

The dual of SR denoted by ECR is

\[
ECR = \begin{cases} 
\min \vec{l}[E] \cdot \vec{x} \\
A^T \cdot \vec{x} \geq \vec{l}_n \\
\vec{x} \geq \vec{0}[E]
\end{cases}
\]

where, this problem is denoted by ECR in order to indicate that it is the relaxed version of the following problem EK, known as the minimum edge covering which is polynomial (\(\mathcal{P}\)).
have their respective constraint sets non empty, they have the same optimal value. So the following inequalities hold: \( u(S) < u(SR) = u(ECR) < u(EC) \)
Proof: From schema (1), the implication \( v(S) = v(EC) \implies v(S) = v(SR) \) becomes obvious. To obtain the converse, we will use a duality argument. Let us revisit the two dual programs SR and ECR. We consider a graph \( G = (V, E) \) with edge-vertex incidence matrix \( A \).

The primal-dual necessary and sufficient optimality conditions for SR and ECR can be expressed as follows: let \((\hat{x}, \hat{y}) \in \mathbb{R}^n \times \mathbb{R}^{\mid E\mid}\); then, the fact that \( \hat{x} \) is a solution of SR and, simultaneously, that \( \hat{y} \) is a solution of ECR, is equivalent to the following conditions:

\[
\begin{align*}
A \cdot \hat{x} & \leq I_{\mid E\mid} & \text{(i)} \\
\hat{x} & \geq 0 \quad & \text{(ii)} \\
A^T \cdot \hat{y} & \geq I_n \quad & \text{(iii)} \\
\hat{y} & \geq 0 \quad & \text{(iv)} \\
\hat{x}_i > 0 & \implies \sum_{j \in \text{adj}(i)} \hat{y}_j = 1 & \text{(v)} \\
\hat{y}_j > 0 & \implies \sum_{i \in \text{extr}(j)} \hat{x}_i = 1 & \text{(vi)}
\end{align*}
\]

where \( \text{adj}(i) \) is the set of the edges adjacent to vertex \( i \) and \( \text{extr}(j) \) is the set of the endpoints of edge \( j \).

If we suppose that \( G \) admits the property \( v(S) = v(SR) \), then there exists a solution \( \tilde{x} \) of SR having only \((0, 1)\) coefficients (\( \tilde{x} \) is the characteristic vector of a maximum independent set \( \tilde{S} \) of \( G \)). Let us now consider a solution \( \tilde{y} \) of ECR (\( \tilde{y} \) having real coefficients). The pair \((\tilde{x}, \tilde{y})\) satisfies the optimality conditions (i) \( \div \) (vi). Let us notice that condition (vi) means that \( S \) "covers" ("touches") all edges having a non-zero value in \( \tilde{y} \).

Consider the bipartite graph \( B(\tilde{S}) = (\tilde{S}, \tilde{V} \setminus \tilde{S}, \tilde{E}') \), with \( e = xy \in \tilde{E}' \) if and only if \( x \in \tilde{S} \) (\( y \notin \tilde{S} \)), or \( x \in \tilde{S} \) and \( e \in E \). The edge-vertex incidence matrix \( A' \) of \( B(\tilde{S}) \) is obtained from \( A \) by deleting the rows of \( A \) corresponding to edges linking vertices of \( \tilde{V} \setminus \tilde{S} \). It is trivial to verify that \( \tilde{S} \) remains an independent set in \( B(\tilde{S}) \), and since it "touches" all the edges of \( \tilde{E} \) having in \( \tilde{y} \) non-zero values, these edges are, by definition, contained also in \( \tilde{E}' \); so, the projection \( \tilde{y} \) of \( \tilde{y} \) in \( \mathbb{R}^{\mid E\mid} \) (we
For the KE graphs where \( v(S) = v(EC) = v(SR) \), it is natural to suppose that \( S \) can be solved in polynomial time; but, although determining the stability number is almost trivial\(^2\), it does not appear evident how we can deduce directly an independent set from an edge covering. In fact, such a solution constitutes, by values equality, a solution of ECR starting from which one can deduce all solutions of its dual program SR. Solving \( S \) becomes then, searching in the polytope of the solutions of SR one solution with \((0,1)\) coefficients, which exists if the graph is KE. But this last step is not a priori simple since, as we show in proposition 1, in the general case, searching for a \((0,1)\) point in a polytope is NP-complete.

**Proposition 1.** Deciding if an integer-linear program and its relaxed version have the same optimal values is NP-complete.

**Proof:** The reduction is from the Hamiltonian circuit problem (HC)\(^3\), which can be expressed in terms of an integer-linear program as follows:

\[
\text{HC} = \left\{ \begin{array}{l}
\max \sum_{i=1}^{n} \bar{t}_i \cdot \bar{x}_i \\
A \cdot \bar{x}_i \geq \bar{x}_{i+1} \quad i \in \{1, \ldots, n\} \\
A \cdot \bar{x}_i \geq \bar{x}_{n} \quad (i) \\
\bar{t}_i \cdot \bar{x}_i \leq \bar{t}_n \quad i \in \{1, \ldots, n\} \quad (ii) \\
\sum_{i=1}^{n} \bar{x}_i \leq \bar{t}_n \quad i \in \{1, \ldots, n\} \quad (iii) \\
\bar{x}_i \in \{0,1\}^n \\n\end{array} \right.
\]

where \( A \) is the vertex-vertex incidence matrix of the graph-instance \( G \) of HC and \( \bar{x}_i, i = 1, \ldots, n \), is a sequence of \( n \) characteristic vectors of a vertex-set.

In the above program, constraints (iii) and (v) imply that each one of the vectors \( \bar{x}_i, i = 1, \ldots, n \), represents one or zero vertices; constraint (i) signifies that if \( \bar{x}_{i+1} \) and \( \bar{x}_i \) represent two vertices, then the represented vertices are neighbours (linked by an edge); so does constraint (ii); in both last cases, if one of the implied vertices has all of its components equal to zero, it is the one of the righthand side of the inequality, and, on the other hand, if both vectors have all of their components equal to zero, then the corresponding constraint is true. The sequence of vectors \( (\bar{x}_1, \ldots, \bar{x}_n) \) is of the form \( (\bar{x}_{k-1}, \ldots, \bar{x}_{k}, \bar{0}_n, \ldots, \bar{0}_n) \), where \( k \in \{0, \ldots, n\} \) and \( \bar{x}_i, i \leq k \), represents a vertex; so, in the case where \( k \neq 0 \), the sequence \( (\bar{x}_1, \ldots, \bar{x}_n) \) represents a path which, in the case \( k = n \), stops on a neighbour of the first vertex of the path. Finally, constraint (iv) means that the path is elementary.

So, if the optimal value of HC is equal to \( n \), then \( G \) is Hamiltonian and, moreover, every optimal solution constitutes a Hamiltonian circuit.
3 A generalization of the König-Égerváry graphs

Throughout this section, we consider the weighted version of $S$, i.e., we consider non-negative weights on the vertices of the graphs; the objective is then to find a maximum-weight independent set, where the weight of an independent set is the sum of the weights on its vertices.

Let us denote by $S_g$ a general instance of the weighted version of $S$ represented by a graph $G = (V, E)$ of order $n$ and size $m$, with $m = 2^k n$ for some $k \geq 0$ and $E'$ the empty graph, from which $E$ is the graph of which both the weights of $V$...
begin
   $S \leftarrow \emptyset$;
   $G_r \leftarrow G$;
   find a solution $\tilde{y}$ of ECR$_d$ in $G$
   
   while $V_r \neq \emptyset$ do
      determine $\tilde{y}$ by projection of $\tilde{y}$ on the space corresponding to $E_r$;
      select an $x_0 \in V_r$, such that $x_0$ is not isolated;
      $\ell(x_0) \leftarrow c$;
      $LV_r \leftarrow \{x_0\}$;
      LABEL($G_r, C, LV_r$);
      if $\neg$TEST($G_r, C, LV_r, \tilde{y}$) then
         begin
            $\ell(x_0) \leftarrow s$;
            $LV_r \leftarrow \{x_0\}$;
            LABEL($G_r, C, LV_r$)
         end;
      $V_r \leftarrow V_r \setminus LV_r$;
      $E_r \leftarrow E_r \setminus \Gamma(LV_r)$;
      $S \leftarrow S \cup \{x_0 \in LV_r : \ell(x_0) = s\}$
   end.
end.

Algorithm 1: Weighted independent set algorithm. We denote by $\ell$ a labelling of the vertices of $G$, that is a function $V \rightarrow \{c, s\}$.

begin
   $E_{test} \leftarrow LV_r$;
   for all $x \in LV_r$ do
      if $\ell(x) = c$ and $\Gamma_{CNr}(x) \not\subseteq LV_r$ then
         for all $y \in \Gamma_{CNr}(x)$ such that $y \notin LV_r$ do
            $\ell(y) \leftarrow s$;
            $LV_r \leftarrow LV_r \cup \{y\}$
         od
      fi
      if $\ell(x) = s$ and $\Gamma_{Er}(x) \not\subseteq LV_r$ then
         for all $y \in \Gamma_{Er}(x) \setminus y \notin LV_r$ do
            $\ell(y) \leftarrow c$;
            $LV_r \leftarrow LV_r \cup \{y\}$
         od
      fi
   od
   if $LV_r \neq E_{test}$ then LABEL($G_r, C, LV_r$) fi
end;

Procedure 1. LABEL.
begin
  TEST ← true
  if $\sum_{v, j(v) = s} a_v \neq a^* \cdot y^*$ then TEST ← false
  else
    delete all the edges of $G_r$ incident to at most one vertex $x$ such that $\ell(x) = s$;
    update all the degrees in $V_r$;
    if $\exists v \in V_r$ with $|I_{E_r}(v)| \neq 0$ then TEST ← false
  fi
end;

Function 1. TEST.

Proof: Consider a graph $G = (V, E)$ of order $n$ belonging to $G_d$; we index by $k$, $k = 1, \ldots, K$,
the iterations of the while loop of algorithm 1, and we denote by $G_r^k = (V_r^k, E_r^k)$, $LV_r^k$ and
$LG_r^k = (LV_r^k, LE_r^k)$, the "current graph" on which iteration $k$ operates, the vertices of $V_r^k$
labelled at the end of the iteration $k$ and the sub-graph of $G_r^k$ induced by $LV_r^k$, respectively;
finally, let us notice that if $k < K$, then $G_r^{k+1}$ is the sub-graph induced by the vertex-set
$V_r^k \setminus LV_r^k$.

The proof of the theorem is essentially based upon the intermediate result expressed by the
following lemma 1.

Lemma 1. The graph $G_r^k$ is a subgraph of $G$ belonging to $G_d$; moreover, for $k < K$, a maximum-
weight independent set $S_r^k$ of $G_r^k$ can be obtained by setting $S_r^k = \{x \in LV_r^k : \ell(x) = s\} \cup S_r^{k+1}$, where $S_r^{k+1}$ is a maximum-weight independent set of $G_r^{k+1}$ and $k$ is the mapping $V_r^k \mapsto \{c, s\}$ computed by algorithm 1.

Proof: (lemma 1.) Let us prove by induction that $G_r^k \in G_d$.
First, we shall prove that, $\forall k$, $G_r^k$ satisfies the property: $\Gamma_C(v) \subseteq V_r^k$, $\forall v \in V_r^k$. For $k = 1$, since $G_1^1 = G$ and $C \subseteq E$, the property is obvious. At this, let us suppose the truth of the property for $k < K$ (we recall that algorithm stops after the $K$th iteration). The set $LV_r^k$ is constructed, during iteration $k$, starting from a singleton and following rule (ii) given above in the description of algorithm 1; so, in the same way, if $v \in V_r^k \setminus LV_r^k$, then $\Gamma C(v) \subseteq V_r^k \setminus LV_r = V_r^{k+1}$ and the property remains true for the induction step $k + 1$.

In order to complete the proof of the fact that, for a given $k$, the graph $G_r^k$ is in $G_d$, we shall prove the following lemma 2.

Lemma 2. Let $G = (V, E)$ be a graph of the class $G_\delta$, $y$ be an optimal solution of ECR$_\delta$ on $G$
and $C$ be as defined previously; let $\tilde{G} = (\tilde{V}, \tilde{E})$ be a subgraph of $G$ induced by $\tilde{V} \subseteq V$ such that,
for all $e = ij \in C$, either $(i, j) \in \tilde{V} \times \tilde{V}$, or $(i, j) \in (V \setminus \tilde{V}) \times (V \setminus \tilde{V})$. Then, $\tilde{G} \in G_d$; moreover, in this case, the tracks of $\tilde{y}$ and $\tilde{x}$ on $\tilde{V}$ are solutions of ECR$_\delta$ and $S_\delta$ on $\tilde{G}$, respectively.

Proof: (lemma 2.) Let $\tilde{x}$ be an optimal solution of $S_\delta$ and, consequently, of $SR_\delta$ ($G \in G_d$); let
us rewrite the primal-dual optimality conditions for the dual linear programs $SR_\delta$ and ECR$_\delta$:
let $(\tilde{x}, \tilde{y}) \in R^n \times R^{\mid E\mid}$; then, the fact that $\tilde{x}$ is a solution of $SR_\delta$ and $\tilde{y}$ is a solution of ECR$_\delta$ is
equivalent to the following conditions:

\[
\begin{align*}
A \cdot \tilde{\mathbf{z}} &\leq \mathbf{\tilde{1}}_{|E|} & (i) \\
\tilde{\mathbf{z}} &\geq 0 & (ii) \\
A^T \cdot \tilde{\mathbf{y}} &\leq \tilde{\mathbf{a}} & (iii) \\
\tilde{\mathbf{y}} &\geq 0 & (iv) \\
\hat{x}_i &> 0 \Rightarrow \sum_{j \in \text{adj}(i)} \hat{y}_j = a_i & (v) \\
\hat{y}_j &> 0 \Rightarrow \sum_{i \in \text{ext}(j)} \hat{x}_i = 1 & (vi)
\end{align*}
\]

where \(\text{adj}(i)\), \(\text{ext}(j)\) and \(A\) are defined as previously.

Let us consider a sub-graph \(\tilde{G} = (\tilde{V}, \tilde{E})\) of \(G\) satisfying the hypotheses of lemma 2; its edge-vertex adjacency matrix is obtained from \(A\) by taking into account only the rows and columns corresponding to the edges and vertices of \(\tilde{G}\), respectively. Let \(\tilde{\mathbf{z}} \in \mathbb{R}^{|\tilde{V}|}\) such that, \(\forall v \in \tilde{V}\), \(\tilde{x}_v = \tilde{\mathbf{z}}_v\) and \(\tilde{\mathbf{y}} \in \mathbb{R}^{|\tilde{E}|}\), such that \(\forall e \in \tilde{E}\), \(\tilde{y}_e = \tilde{y}_e\) the projections of \(\tilde{\mathbf{z}}\) and \(\tilde{\mathbf{y}}\) on the characteristic spaces of \(\tilde{V}\) and \(\tilde{E}\), respectively; we also define \(\tilde{\mathbf{z}} \in \mathbb{R}^{|\tilde{V}|}\) such that, \(\forall v \in \tilde{V}\), \(\tilde{a}_v = a_v\), the projection of vector \(\tilde{\mathbf{a}}\) (it is easy to see that \(\tilde{\mathbf{z}} \in \{0, 1\}^{|\tilde{V}|}\)).

We shall show that the pair \((\tilde{\mathbf{z}}, \tilde{\mathbf{y}})\) satisfies the primal-dual optimality conditions (i) + (vi) (for \(\text{SR}_G\) and \(\text{ECR}_G\), respectively) in the graph \(\tilde{G}\) of edge-vertex incidence matrix \(\tilde{A}\).

In fact, \(\tilde{\mathbf{z}}\) is the characteristic vector of an independent set of \(\tilde{G}\) because \(\tilde{\mathbf{z}}\) corresponds to an independent set of \(G\) and, moreover, \(\tilde{G}\) is a sub-graph of \(G\); so, \(\tilde{A} \cdot \tilde{\mathbf{z}} \leq \mathbf{\tilde{1}}_{|\tilde{E}|}\) and \(\tilde{\mathbf{z}} \geq 0_{|\tilde{V}|}\) (conditions (i) and (ii)). It is also easy to obtain \(\tilde{\mathbf{y}} \geq 0_{|\tilde{E}|}\) (condition (iv)). Moreover, vector \(\tilde{\mathbf{y}}\) satisfies \(\tilde{A} \cdot \tilde{\mathbf{y}} \geq \tilde{\mathbf{a}}\) (condition (iii)).

Plainly, let us consider one of these constraints; it corresponds to a vertex \(v \in \tilde{V}\); let an edge \(e \in \tilde{E}\) incident to \(v\); if \(\tilde{y}_e > 0\), then, by definition of \(C\), \(e \in C\) and hence, since \(v \in \tilde{V}\), by the hypotheses of lemma 2, \(e \in \tilde{E}\); so, the sum of the values \(\tilde{y}_e\), \(e\) incident to \(v\) in \(G\) equals the sum of values \(\tilde{y}_e\), \(e\) incident to \(v\) in \(G\).

Let us now verify the exclusion conditions (v) and (vi). The fact that \(\tilde{x}_v > 0\) means also that \(\hat{x}_v > 0\), hence the corresponding constraint of \(\text{ECR}_G\) is saturated, so does the corresponding constraint of \(\text{ECR}_G\) in \(\tilde{G}\) given that the sums of the values of the edges incident to \(v\) in both \(G\) and \(\tilde{G}\) are equal, thus condition (v) is satisfied. Finally, if \(\tilde{y}_e > 0\), then \(\hat{y}_e > 0\) and consequently, the sum of the values \(\hat{x}_v\), where \(v\) endpoint of \(e\), is equal to 1; since both these endpoints belong to \(\tilde{V}\), condition (vi) holds also for \(\tilde{G}\).

As a conclusion, vectors \(\tilde{\mathbf{z}}\) and \(\tilde{\mathbf{y}}\) are solutions of \(\text{SR}_G\) and \(\text{ECR}_G\), respectively, defined on \(\tilde{G}\); moreover, \(\tilde{\mathbf{z}}\) has \((0,1)\) coordinates; so, \(\tilde{G} \in \mathcal{G}_d\) and this completes the proof of lemma 2.

We now continue the proof of lemma 1.

By lemma 2, for a given \(k\), \(G^k_r\) is an element of \(\mathcal{G}_d\); so, the first part of the lemma is proved.

Let us notice that, from the proof of lemma 2, we can deduce also that \(G^k_r \in \mathcal{G}_d\) and, moreover, the intersections of a maximum-weight independent set of \(G\) with \(V^k_r\) and \(LV^k_r\), constitute maximum-weight independent sets for \(G^k_r\) and \(G^k_r\), respectively; if we denote also by \(\tilde{y}_r^k\) and \(\tilde{y}_r^k\) the projections of \(\tilde{y}\) on the spaces corresponding to \(E^k_r\) and \(E^k_r\), respectively, the graphs \(G^k_r\) and \(G^k_r\) have weighted-stability numbers equal to \(\tilde{\alpha}_{V^k_r} \cdot \tilde{y}_r^k\) and \(\tilde{\alpha}_{LV^k_r} \cdot \tilde{y}_r^k\), respectively.

Given a labelling \(\ell : V^k_r \mapsto \{c, s\}\) of \(G^k_r\), we call it a "good labelling" if there exists a maximum-weight independent set of \(G^k_r\) containing all of the vertices labelled by \(s\) and none of the ones labelled by \(c\). Let us remark that procedure LABEL proceeds only by necessary conditions, in the sense that, for a fixed maximum-weight independent set, if a vertex belongs to it, then its neighbours belong to the associated minimum-weight vertex covering; on the other hand, if a vertex \(v\) belongs to the minimum-weight vertex covering associated to the fixed maximum-weight independent set, then optimality condition (vi) in the proof of lemma 2 imposes that \(\Gamma_C(v)\) is
included in this (fixed) independent set. These two properties of procedure LABEL make that the procedure completes a "good labelling" by producing a "good labelling". So, in the case where the initial labelling, assigning label c to a vertex \( x_0 \) and no label to any other vertex (see algorithm 1) is a "good labelling", then the labelling completing this initial one is also a "good labelling". In this case, the vertices labelled by s constitute, in \( G_r^k \), the track of a maximum-weight independent set of \( G \), hence a weighted independent set of value \( d_{|LV|} \cdot L \), this fact being tested by function TEST. Consequently, since \( LG_r^k \in \mathcal{G}_r \), by lemma 2, the value of this function is true if and only if the vertices labelled by s constitute a maximum-weight independent set in \( LG_r^k \). If this is not true, then the initial labelling was not a "good labelling".

i.e., \( x_0 \) belongs to every maximum-weight independent set. So, the labelling assigning to \( x_0 \) the label s and assigning no label to any other vertex, is a good labelling so does, consequently, the completed one; hence, with the same arguments, the vertices labelled by s always constitute a maximum-weight independent set of \( LG_r^k \). To summarize, after one or two calls of procedure LABEL by algorithm 1 during an iteration of its while loop, the vertices labelled by s after the last call constitute always a maximum-weight independent set of \( LG_r^k \) of value \( d_{|LV|} \cdot L \). On the other hand, if \( k < K \), a maximum-weight independent set of \( G_r^{k+1} \) has value \( d_{|V|} \cdot L \).

so, since the disjoint union of \( LV_r^k \) and \( V_r^{k+1} \) equals \( V_r^k \), the union of these weighted independent sets (in \( G_r^k \) and \( G_r^{k+1} \)) has value \( d_{|LV|} \cdot L + d_{|V|} \cdot \frac{y_r}{y_r} = d_{|V|} \cdot \frac{y_r}{y_r} \), which is the value of a maximum independent set in \( V_r^k \). So, to prove that this union constitutes a maximum-weight independent set for \( G_r^k \), it suffices to show that this union constitutes an independent set for \( G_r^k \).

In fact, by the way the vertex-labelling is performed (if a vertex is labelled by s, then all of its neighbours are marked by c and introduced in \( LV_r^k \)), the vertices labelled by s at the end of iteration \( k \) are not linked, in \( G \), to any vertex of the set \( V_r^{k+1} \); so, the union of an independent
4 König-Egerváry graphs revisited

In the case where $a = I_n$, the results of section 3 mean that one can decide, in polynomial
time, if a given graph $G$ is KE and, if this is the case, determine, always in polynomial time, a
maximum independent set of $G$.

In this case, the constraint $\tau(G) = m$, where $\tau(G)$ and $m$ are the sizes of a minimum vertex
covering and of a maximum matching of $G$, respectively, imposes that the set of the exposed
vertices of $G$, with respect to a given matching $M$, are included in at least one of the maximum
independent sets of $G$ (let us denote by $S$ such an independent set); moreover, for every edge
of $M$, exactly one of its endpoints is included in the minimum vertex covering $V \setminus S$.

Let us notice that, in this case, one could, slightly, simplify algorithm 1 by initializing the
solution to the exposed vertices in order to obtain, after one application of procedure LABEL,
a graph admitting a perfect matching (the sub-graph of $G$ induced by the set of the matching
edges endpoints); then, the set $C$ defined and used in section 3 is exactly the perfect matching
of the reduced graph.

In this case, the step of algorithm 1, consisting of solving a linear program in order to find a
solution of ECR, becomes to find a maximum matching, the complexity of this step becoming
$O(n^{2.5})$ ([7]). But, in any case, the complexity of the so simplified algorithm remains of $O(n^3)$.

So, the following theorem summarizes the discussion of this small section.

**Theorem 4.** There exists an $O(n^3)$ algorithm deciding if a given graph $G$ is KE and, if so,
determining a maximum independent set of $G$.

We find in theorem 4 the well-known result of [5]. In any case, let us notice that, in comparison
with the corresponding theorem of [5], both algorithm 1 and the proof of theorem 2 are much
less complicated and simpler (for identical algorithmic complexities).

5 Some independent set approximation and exact polynomial results "inspired" from König-Egerváry graphs

Let us consider a minimum vertex cover $C^*$ and the corresponding maximum independent set $S^*$
in a graph $G$. Let us also suppose that, given a matching $M$, there are $f$ matching edges such
that both their endpoints belong to $C^*$, for the remaining ones, one of their endpoints belonging
independent set problem; in the first case, these results are exact, while in the second case, the given results are approximation ones.

**Proposition 2.** Consider a graph $G = (V, E)$ such that $0 \leq \tau(G) - m = f + g \leq \kappa$ (where $m$ is the cardinality of a maximum matching $M$). Then, (i) if $\kappa$ is a fixed positive integer constant, there exists an exact polynomial algorithm for maximum independent set problem in $G$; (ii) otherwise, there exists a polynomial time approximation algorithm (having $\kappa$ among its input parameters) providing an independent set of cardinality at least equal to $\lceil \sqrt{2(\log \kappa + 1)} \rceil$. 
Part II
The approximability behaviour of some
begin
    compute a maximum matching $M$ in $G$;
    $C \leftarrow T[M]$
end;

**Procedure 2.**

From the above expressions, we deduce

$$\tau'(G) \leq \left(1 - \frac{2}{3}\varepsilon\right)n$$

and hence one can obtain immediately an independent set on $G$ of cardinality

$$\alpha'(G) = n - \tau'(G) \geq \frac{2}{3}cn.$$ (4)
(5)

Consequently, $\mathcal{A}$ (provided with a set-difference instruction) constitutes a polynomial algorithm for $S_3$, always guaranteeing a ratio $\alpha'(G)/\alpha(G) \geq \alpha'(G)/n \geq (2/3)\varepsilon$, this ratio being a (universal) constant. So, on the hypothesis that $S_3$ is not constant-approximable in polynomial time, such a polynomial time approximation algorithm $\mathcal{A}$ cannot exist for VC (unless $P = NP$). 

### 6.2 The constant-approximability of $S_3$ would induce an improvement on vertex cover's approximation ratio

In order to prove the conditional result of theorem 5 of section 6.2.2, we present in section 6.2.1, a polynomial time approximation algorithm for VC (algorithm 2). Moreover, we suppose that there exists a polynomial time approximation algorithm $\mathcal{A}$ for $S_3$ with a fixed positive constant approximation ratio $\rho$. In section 6.2.2, we show that, under this hypothesis, algorithm 2 guarantees an approximation ratio strictly smaller than 2 for VC.

#### 6.2.1 An algorithm for vertex covering and its properties

We introduce and discuss now three different procedures for finding a vertex covering in a graph $G$, which will be then exploited in a more general algorithm (algorithm 2 presented at the end of this section). In fact, as we shall see, algorithm 2 calls algorithm 1 and the three procedures presented in what follows and chooses the smallest among the produced solutions.

All the three procedures and algorithms 2 and 1 have as input a graph $G$ (without loss of generality, we can suppose that $G$ is connected) and output a vertex covering for $G$. In what follows, by $C$ and $S$, respectively, we shall denote a vertex covering and the independent set associated with $C$, i.e., $S = V \setminus C$.

First, procedure 2, the maximum matching algorithm ([?]; this is the most-known approximation algorithm for VC), is called.

In the case where $M$ (the matching computed by procedure 2) is perfect, procedure 3 is called to provide a solution for $G$. procedure 3, is a simple procedure calling the hypothetical constant-ratio approximation algorithm $\mathcal{A}$, and then taking the complement of a solution provided by $\mathcal{A}$.

Finally, procedure 4 treats the case where $G$ admits a non-perfect maximum matching. Let $M$ be a maximum matching of $G$, with $|M| = m$ and suppose that $M$ is not perfect. Let $S$

\[\text{We suppose that whenever } \mathcal{A} \text{ operates on graphs } G \text{ which are not instances of } S_3, \text{ it stops in polynomial time, delivering either non-feasible solutions or maximal independent sets of cardinality smaller than } \rho n/3.\]
be the independent set derived by procedure 3 when applied to $G' = G[T[M]]$; let $X$ be the set of the exposed vertices of $V$ with respect to $M$, and let $M_1 \subseteq M$ ($|M_1| = m_1$) be the edges of $M$ having one endpoint in $S \cap \Gamma(X)$, where, for a vertex-set $Y \subseteq V$, we denote by $\Gamma(Y)$ the set of vertices of $V \setminus Y$ joined by an edge to at least a vertex of $Y$ (informally speaking, $\Gamma(Y)$ is the set of neighbours of the vertices in $Y$). Let $M_2 = M \setminus M_1$ ($|M_2| = m_2 = m - m_1$); also, let us assume that $T[M_1] \cap S = \{s_1, \ldots, s_{m_1}\}$ and $c_i = m(s_i)$, $i = 1, \ldots, m_1$; let $X_1 = \Gamma(T[M_1] \cap S) \cap X$ and let $X_2 = X \setminus X_1$. Finally, let us note that the set $C$ (output of procedure 4) is initialized at line (3) of the procedure by the output of procedure 3 called at this line and it is completed by the execution of either the consequence then, or the consequence else of procedure 4.

The following lemma brings to the fore an interesting property of procedure 4, in the case where the consequence else of the if clause of procedure 4 is executed; this property is used in the proof of lemma 5 (establishing the correctness of procedure 4) as well as in the proof of theorem 5.

**Lemma 3.** Consider a vertex $v \in S_1 \setminus X$; then, there exists an exposed vertex $x \in X_1$ and an alternating path from $v$ to $x$ starting with $vm(v)$, all edges of this path being included in $E_x$.

**Proof:** From procedure 4 and since $v \notin X$, there exists $l \in \{1, \ldots, |X_1|\}$ such that $v$ is introduced in $S_1$ during the $l$th iteration of the for loop of line (12). We then distinguish two cases: (i) $v \in S_1 \cap C$, and (ii) $v \in S_1 \setminus S$.

For case (i), $v = m(v) - x_l$ is the searched path.

Let us now discuss case (ii). Vertex $v$ is introduced in $S_1$ during an execution either of line (19) (case (j)), or of line (22) (case (jj)), or of line (25) (case (jjj)).

In case (j), the $5$-cycle discovered at line (17) is the cycle $x_l - v - m(v) - c - m(c) - x_l$ (with $m(c) \neq v$) and the searched path is $v - m(v) - c - m(c) - x_l$.

Case (jj) (the case of triangles) is similar to the case (j).

Before considering case (jjj), let us note that since lines (17)-(22) have all been executed, for every vertex $s \in S_1^k$, vertex $m(s) \in C_1^k$. Let us now consider case (jjj). Let us number from 1 to $N$ the executions of line (25) (since we treat case (jjj), $N \geq 1$) and, for all $k \in \{1, \ldots, N\}$, let us denote $s_k$ the vertex introduced in $S_1^k$ during the $k$th execution of line (25); let us denote $S_1^k(k)$ and $C_1^k(k)$, the subsets of $S_1^k$ and $C_1^k$, respectively, resulting from the insertion of $s_k$ in $S_1^k$. If line (25) has been executed at least once, then line (19), or line (22) has also been executed at least once; let us denote by $S_1^0(0)$ and $C_1^0(0)$, the non-empty subsets of $S_1^0$ and $C_1^0$ resulting from the last execution of lines (19), or (22).

Let us now show by induction on $k \in \{0, \ldots, N\}$ that: (a) $\forall v \in S_1^k(k) \cup C_1^k(k)$, $m(v) \in S_1^k(k) \cup C_1^k(k)$ and (b) for every vertex $s \in S_1^k(k)$, there exists an alternating path from $s$ to $x_l$ starting from a matched edge and exclusively containing vertices of $S_1^k(k) \cup C_1^k(k)$.

**Basis:** for $k = 0$, (a) and (b) immediately result from the discussion of cases (j) and (jj).

**Inductive step:** suppose (a) and (b) true for $k < N$; the only newly introduced in $S_1^k(k) \cup C_1^k(k)$ vertices being $s_{k+1}$ and $m(s_{k+1})$, property (a) is obviously satisfied on range $k + 1$; for
begin
(1) compute a maximum matching $M$ in $G$;
(2) $G' = G[T[M]]$
(3) call procedure 3 on $G'$ to obtain sets $C$ and $S$;
(4) determine $M_1$, $M_2$, $X_1$ and $X_2$;
(5) if $m_1 \leq \rho m/3$ then $C \leftarrow C \cup \{T[M_1] \cap S\}$
   else
(6)   $C_2 \leftarrow T[M_2]$;
(7)   $C_1 \leftarrow \emptyset$;
(8)   $S_1 \leftarrow \emptyset$;
(9) order arbitrarily the elements of $X_1$;
(10) let $X_1 = \{x_1, \ldots, x_{|X_1|}\}$ be the resulting ordering
(11) for $l \leftarrow 1$ to $|X_1|$ do
(12)    $C_1[l] \leftarrow \emptyset$;
(13)    $S_1[l] \leftarrow \emptyset$;
(14)    $V_1 \leftarrow T[M_1[x_l]] \cup \{x_l\}$;
(15)    $E_l \leftarrow E(G[V_1])$;
(16) find all 5-cycles $x_l - s_i - c_i - c_j - s_j - x_l$ such that \{s_i, c_i, c_j\} $\subseteq M_1$;
(17) $C_1[l] \leftarrow C_1[l] \cup \{x_l, c_i, c_j\}$;
(18) $S_1[l] - S_1[l] \cup \{s_i, s_j\}$;
(19) find all triangles $x_l - s_i - c_i - x_l$ such that \{s_i, c_i\} $\subseteq M_1$;
(20) $C_1[l] \leftarrow C_1[l] \cup \{x_l, c_i\}$;
(21) $S_1[l] - S_1[l] \cup \{s_i\}$
(22) while $\exists (c_k \in (V_1 \setminus C_1[l]) \land s_i \in S_1[l])$, $c_k \in E_l$, $k \neq i$ do
(23)    $C_1[l] \leftarrow C_1[l] \cup \{c_k\}$;
(24)    $S_1[l] - S_1[l] \cup \{m(c_k)\}$
(25) od
(26) while $\exists s_kc_k \in M_1$, \{s_k, c_k\} $\subseteq V_1$, \{s_k, c_k\} $\cap C_1[l] = \emptyset$ do
(27)    $C_1[l] \leftarrow C_1[l] \cup \{s_k\}$;
(28) $S_1[l] - S_1[l] \cup \{s_k\}$
(29) od
(30) $C_1[l] \leftarrow C_1[l] \cup C_1[l]$;
(31) $S_1[l] - S_1[l] \cup S_1[l]$;
(32) od
(33) $S_1 \leftarrow S_1 \cup (X \setminus C_1)$;
(34) $C \leftarrow C_1 \cup C_2$
(35) fi
end;

Procedure 4.
property (b) on range \( k + 1 \), it suffices to consider the case where \( s_{k+1} \notin C'_1 \) (the opposite case being treated in case (i)); then, from line (23) of procedure 4, \( m(s_{k+1}) \notin C'_1(k) \) and, also, \( \exists s \in S'_1(k) \) such that \( m(s_{k+1})s \in E_l \); from the inductive hypothesis on property (b), there exists an alternating path \( \mu \) from \( s \) to \( x_l \) exclusively containing vertices from \( S'_1(k) \cup C'_1(k) \); on the other hand, from the inductive hypothesis on property (a), \( \{s_{k+1}, m(s_{k+1})\} \notin S'_1(k) \cup C'_1(k) \) and, moreover, \( (S'_1(k) \cup C'_1(k)) \cap \{s_{k+1}, m(s_{k+1})\} = \emptyset \); so, the path \( s_{k+1} - m(s_{k+1}) - \mu \) is the searched alternating path concluding the proof.

To illustrate the property described by lemma 3, let us consider the following example.

**Example 1.** Let us consider the graph of figure 1. Suppose that at line (c) of procedure 4, the cycle \( \pi = x - s_1 - c_1 - c_2 - s_2 - x \) has been detected, where \( x \in X \), \( c_1 \) and \( c_2 \) belong to \( C \) (the vertex covering of \( G' \)), and \( s_1, s_2 \) belong to \( S \) (the independent set of \( G' \) detected by procedure 3).

Once \( \pi \) is detected, \( c_1 \) and \( c_2 \) are added in \( C_1 \) and \( s_1, s_2 \) in \( S_1 \). During the first while loop, the vertex set \( \{c_3, c_4, c_5, c_6\} \) has also been added in \( C_1 \), the insertion of each \( c_i, i = 3, \ldots, 6 \), in \( C_1 \) entailing the insertion of \( s_i = m(c_i), i = 3, \ldots, 6 \), in \( S_1 \). Then, for \( s_6 \), the alternating path claimed by lemma 3 is the path \( \mu = s_6 - c_6 - s_5 - c_5 - s_4 - c_4 - s_3 - c_3 - s_2 - c_2 - c_1 - s_1 - x \); for the rest of the vertices \( s_i, i = 2, \ldots, 6 \), the alternating paths claimed by lemma 3 are the fragments of \( \mu \) starting from \( s_i \), while for \( s_1 \), the alternating path is the path \( s_1 - c_1 - c_2 - s_2 - x \).

Let us now prove another easy lemma concerning procedure 4 and used in the proof of theorem 5 in section 6.2.2.

**Lemma 4.** There does not exist an edge \( uv \in M_1 \) such that there exist \( \{x_i, x_j\} \subseteq X, x_i \neq x_j, \) with \( \{ux_i, vx_j\} \subseteq E \).

**Proof:** Plainly, if the contrary is true, then \( x_i - u - v - x_j \) is an augmenting (alternating) path, contradicting the maximality of \( M \).

A particular case of lemma 4 is that there is no edge \( uv \in M_1 \) such that one of its endpoints, say \( u \), is linked to an exposed vertex \( x_i \in X_1 \) and the other one, say \( v \), is linked to an exposed vertex \( x_j \in X_2 \).

**Lemma 5.** (Correctness of procedure 4.) Procedure 4 finds in polynomial time a vertex covering \( C \) of its input graph \( G \).

**Proof:** Considering the time complexity of procedure 4, it is easy to see that its most "expensive" operation is the instruction of line (17) performed at most \( |X_1| \) times. This operation entails a worst-case complexity of \( n^6 \) (where \( n \) is the order of \( G \)).
Consider now the set $C$ created during the execution of the `then` consequence of the `if` clause of procedure 4.

It is easy to see that since the only uncovered edges are the ones of the form $uv$, where $u \in S \cap T[M_1]$ and $v \in X_1$, then set $S \cap T[M_1]$ suffices to cover them; therefore, $C \cup (S \cap T[M_1])$ constitutes a vertex covering for $G$.

Let us now consider the set $C$ created during the `else` consequence of the `if` clause of procedure 4.

We arbitrarily label by $s_i$, $i = 1, \ldots, m_1$, the vertices of $S \cap T[M_1]$, and by $c_i$ the vertices $m(s_i)$, $i = 1, \ldots, m_1$.

We prove now that set $C_1$ created by the procedure constitutes a vertex covering for $G[T[M_1] \cup X_1]$. In order to do that, we distinguish two families of edges in this graph: (i) edges of $E_l$, $x_i \in X_1$, and (ii) edges $u_i v_j$, where $u_i \in V_l$, $v_j \in V_p$, $\{x_i, x_p\} \subseteq X_1$, $l \neq p$.

(i) We first prove that the subset of $C_1$ created at the $l$th iteration of the `for` loop of the procedure (concerning $x_i \in X_1$) covers all edges in $E_l$.

Let us first point out that $S_l^1 \cap V_l = V_l \setminus C_l^1$. In fact, obviously, $C_l^1 \cap S_l^1 = \emptyset$; line (34) of procedure 4 guarantees that $X \subseteq S_l \cup C_1$; on the other hand, at the $l$th iteration of the `for` loop, $l = 1, \ldots, |X_1|$, if a vertex of $T[M_1]$ is introduced in $C_l^1$, its mate is immediately introduced in $S_l^1$.

Finally, the second `while` loop (lines (27)-(30)) guarantees that all of the edges of $M_1 \cap E_l$ are covered by $C_l^1$. Hence, $S_l^1 \cup C_l^1 \supseteq T[M_1 \cap E_l] \cup X \supseteq V_l$.

Observe that when the execution exits the first `while` loop (lines (23)-(26)), $S_l^1 \subseteq S_l$, and thus $S_l^1$ is an independent set of $G$. Now, let $s_i \in S_l^1$, suppose there exists $s_j \in S_l^1$ such
begin
  call algorithm 1 on \( G \)
  if \( G \) is \( \text{KE} \) then find and store the complement of the provided independent set
  call procedure 2 and store the obtained solution for \( \text{VC} \)
  compute a maximum matching \( M \) in \( G \)
  if \( M \) is perfect then call procedure 3 and store the obtained solution for \( \text{VC} \)
  else call procedure 4 and store the obtained solution for \( \text{VC} \)
fi
choose, between the two candidate solutions, the smallest one
end.

Algorithm 2.

On the other hand, there is no edge between \( X_2 \) and \( T[M_1] \cap S_1 \) because, in the opposite case, the application of lemma 3 would bring to the fore an augmenting path. So, the set \( C \) obtained at the end of the else consequence of the if condition of procedure 4 constitutes a vertex covering for \( G \).

Let us remark here that the case of triangles (lines (20) and (21) of procedure 4) is similar to the case of cycles just examined; so, the proof for triangles is omitted.

We give now an overall specification of algorithm 2, the approximation performance of which is studied in section 6.2.2; we recall that this algorithm uses the hypothetical constant-ratio approximation algorithm \( A \) (directly called by procedure 3) for \( S_3 \).

6.2.2 The main result

Theorem 5. On the hypothesis that algorithm \( A \) is a polynomial time \( p \)-approximation algorithm for \( S_3 \), for \( p < 1 \) a fixed positive constant, algorithm 2 is a polynomial time approximation algorithm for \( \text{VC} \), guaranteeing approximation ratio smaller than \( 2 - (p/6) < 2 \).

Proof: Let \( G = (V, E) \) be a graph of order \( n \), instance of \( \text{VC} \), and \( M \) (\( |M| = m \)) be a maximum matching of \( G \).

Let us consider a minimum vertex cover \( C^* \) and the corresponding maximum independent set \( S^* \) in \( G \), i.e., \( S^* = V \setminus C^* \). As previously, let us also suppose that there are \( f \) matched edges (already called "dissymmetric"), such that both their endpoints belong to \( C^* \), for the remaining ones, one of their endpoints belonging to \( C^* \) and the other one to \( S^* \); let us denote by \( F \) the set of "dissymmetric" edges (\( |F| = f \)); given \( X \), the set of the exposed vertices of \( G \) with respect to \( M \) (when \( M \) is a perfect matching, \( X = \emptyset \)), let us denote by \( X_C \) (\( |X_C| = g \)) and \( X_S \), the subsets of \( X \) belonging to \( C^* \) and \( S^* \), respectively (of course, \( X = X_C \cup X_S \)). In fact,

\[
|C^*| = \tau(G) = m + (f + g)
\]

and, consequently, \( |S^*| = \alpha(G) = n - m - (f + g) \).

In order to prove the theorem, we distinguish three cases for the quantity \( f + g \). We first study the case (a) \( f + g = 0 \); next, we study the case (b) \( f + g \geq m/3 \); finally, we study the case (c) \( 0 < f + g \leq m/3 \).

(a) \( f + g = 0 \).

This is the case where algorithm 1 is called for \( \text{VC} \). As we have seen in part I, in this case \( \text{VC} \) admits a polynomial time algorithm of approximation ratio equal to 1. So, for case (a) we have

\[
\frac{\tau'(G)}{\tau(G)} = 1.
\]

19
(b) $f + g \geq m/3$.
In this case, procedure 2 is used as an approximation algorithm for VC.
We have then $\tau'(G) = 2m$. From the expression (6) and the one for $\tau'(G)$, we obtain the following approximation ratio: $\tau'(G)/\tau(G) = 2m/[m + (f + g)]$. Then, for $(f + g) \geq m/3$, we obtain immediately an approximation ratio

$$\frac{\tau'(G)}{\tau(G)} \leq \frac{3}{2}$$  \hspace{1cm} (8)

for the maximum matching procedure 2, whenever used to solve approximately VC.

(c) $0 < (f + g) \leq m/3$.
In this case, we distinguish two subcases (c.1) and (c.2) depending on whether $M$ is perfect or not.

(c.1) **M is perfect** ($g = |X_S| = |X_C| = |X| = 0$).
Then, $n = 2m$ and procedure 3 is used to obtain an approximate solution for VC.
The expression (6) for $\tau(G)$ gives $\tau(G) = m + f \leq 4m/3$; consequently, $\alpha(G) \geq 2m - (4m/3)$;
so $\alpha(G) \geq 2m/3 = n/3$.
Thus, $G$ is an instance of $S_3$ and, when $A$ operates on $G$, gives $\alpha'(G) \geq \rho n/3$. Then, the following inequalities hold for $G$: $\alpha'(G) \geq \rho n/3$, $\tau'(G) = n - \alpha'(G) \leq [1 - (\rho/3)]n$, $\tau(G) \geq m = n/2$ and, consequently,

$$\frac{\tau'(G)}{\tau(G)} \leq 2 - \frac{2\rho}{3} < 2.$$  \hspace{1cm} (9)

(c.2) **M is not perfect** ($X \neq \emptyset$).
This is the case where procedure 4 is called to solve VC (recall that in proposition 5 it is proved that procedure 4 feasibly solves VC). Consider the graph $G' = G[V \setminus X]$; $M$ is perfect for $G'$. Obviously, since $\alpha(G') \geq m - f \geq 2m/3 = |V(G')|/3$, $G'$ is an instance of $S_3$. The call of procedure 3 to $G'$ (performed by procedure 4) initializes sets $C$ and $S$ and allows the computation of $M_1$, $M_2$, $X_1$ and $X_2$.

With respect to $M_1$, we consider two cases, namely $m_1 \leq \rho m/3$ (case (c.2.1)) and $m_1 \geq \rho m/3$ (case (c.2.2))

(c.2.1) $m_1 \leq \rho m/3$. 

to $X_S$) forming triangles with the same edge of $M_1 \cap F$. Since we have $f$ edges in $F$ and $g$ vertices in $X_C$, then there are less than $f + g$ exposed vertices introduced in $C_1$ at line (21) of the procedure.

Let us now see how many exposed vertices have been introduced in $C_1$ because of the cycles discovered at line (17) of the procedure. First, let us define the intersection of two such cycles to be the set of their vertices in common. The particular form of these cycles (they contain two matched edges) induces that their intersection could only arise either on an exposed vertex, or on an exposed vertex and the endpoints of a matched edge. The arguments: it is easy to see that two such cycles cannot be intersected on a set of vertices containing only one endpoint of a matched edge; on the other hand, if there exist two cycles $x_i - s_{ij} - m(s_{ik}) - m(s_{ih}) - s_{ij} - x_i$ and $x_j - s_{jk} - m(s_{ik}) - m(s_{ih}) - s_{ij} - x_j$, $x_i \neq x_j$ (intersected only on the endpoints of the (matched)
fixed positive constant ε.

To prove proposition 5, it suffices to replace, in the part (i) of theorem 5, (3/2) − ε by 2 − ε; then, expression 3 gives r(G) ≤ n/2, expression 4 gives r′(G) ≤ [1 − (ε/2)n], from expression 5, we get α′(G) = n − r′(G) ≥ (2/3)εn. Consequently, the hypothetical algorithm A constitutes a polynomial algorithm for S_2 guaranteeing always α′(G)/α(G) ≥ α′(G)/n ≥ ε/2, a contradiction, since ε/2 is a (universal) constant.

7 Mathematical programming and maximum independent set

7.1 Convex programming and maximum independent set

The (maximization) convex programming problem considered here is defined as follows:

\[
\text{CPM}(\kappa) = \max \sum_{i \in \{1, \ldots, \kappa\}} f(x_i)
\]

where \(P\) is a polytope defined by a finite number of constraints, and \(f\) belongs to a family \(F\) of functions increasing in [0, 1], with \(f(0) = 0\) for every \(f \in F\), verifying the property

\[
\inf_{f \in F} \left\{ \frac{f(\frac{1}{2})}{f(1)} \right\} \in \left[ 0, \frac{1}{\kappa} \right], \quad \kappa \geq 2.
\] (12)

The following theorem is the main result of this section.

**Theorem 7.** Let \(\kappa \geq 2\) be such that \(S_\kappa\) does not admit a polynomial time algorithm guaranteeing a maximal independent set greater than \(\rho n\) for a fixed positive constant \(\rho < 1\). Then, on the hypothesis that \(P \neq NP\), there does not exist a polynomial time approximation algorithm for CPM(\(\kappa\)) guaranteeing an approximation ratio greater than \(\kappa \inf_{f \in F} \{f(1/2)/f(1)\}\).

**Proof:** Let us suppose the existence of an approximation algorithm \(A\) for CPM(\(\kappa\)), with a constant ratio \(\rho < 1\). Given an instance of CPM(\(\kappa\)), let us denote by \(\hat{v}\) the value of the solution \(\hat{x}\) found by \(A\), and by \(v^*\) the value of the optimal solution \(x^*\); of course, \(\hat{v} \geq \rho v^*\).

Let us also consider an instance of the maximum independent set problem, in other words a graph \(G\) of order \(n\) with edge-vertex matrix \(A\). This instance defines a family of instances of CPM(\(\kappa\)) for which \(P = \{\hat{x} \in R^n : A \cdot \hat{x} \leq \hat{1}, \hat{x} \geq 0\}\). We denote this family by IPM(\(\kappa\)) and we express it in terms of a nonlinear program with linear constraints as follows:

\[
\text{IPM}(\kappa) = \max \sum_{i \in \{1, \ldots, n\}} f(x_i)
\]

\[
A \cdot \hat{x} \leq \hat{1}, \quad \hat{x} \geq 0
\]

Of course, IPM(\(\kappa\)) being a particular instance of CPM(\(\kappa\)), it can be solved approximately within a ratio \(\rho\). Let us now consider an instance of IPM(\(\kappa\)) and let us denote by \(v^*(f)\) its optimal value, and by \(\hat{v}(f)\) the value of the approximate solution found by \(A\). Let also \(\alpha(G)\) be the stability number of \(G\), let \(S^*\) be a maximum independent set of \(G\), and \(\hat{x}(S^*)\) the corresponding vector.

Since, \(\forall f \in F, \hat{x}(S^*)\) is feasible for IPM and since the objective value of this vector is \(f(1)\alpha(G)\), we have, \(\forall f \in F\),

\[
\hat{v}(f) \geq \rho v^*(f) \geq \rho f(1)\alpha(G).
\] (13)

On the other hand, let us consider the solution \(\hat{x}(f)\) of IPM(\(\kappa\)). We then consider the set \(\hat{S} = \{i \in \{1, \ldots, n\}, \hat{x}_i(f) > 1/2\}\), where \(x_i\) is the \(i\)th component of vector \(\hat{x}\). It is easy to
see that \( S \) constitutes an independent set for \( G \). Plainly, the facts: \( A \cdot \tilde{x}(f) \leq 1 \) and \( \tilde{x}(f) \geq 0 \) imply, on the one hand, that \( \tilde{x}_i(f) \in [0, 1] \), for all \( i \in \{1, \ldots, n\} \) (\( A \) is a 0–1 array), and, on the other hand, that two adjacent vertices have positive values, the sum of which cannot exceed 1;
Theorem 8. Let $\kappa \geq 2$ such that $S_\kappa$ does not admit a polynomial time algorithm guaranteeing a (universally) constant approximation ratio. Then, unless $P = NP$, there does not exist a polynomial time approximation algorithm for CPm(\kappa) guaranteeing an approximation ratio smaller than $[\kappa/(\kappa - 1)]\sup_{f \in \mathcal{F}} \{f(1/2)/f(1)\}$.

Proof: We use the same notations as in the proof of theorem 7. Let us suppose the existence of a polynomial time approximation algorithm $A$ with a fixed constant approximation ratio $\rho$ for CPm. Then, to every instance of the maximum independent set problem, we associate the following family of instances of CPm:

$$IP_m = \begin{cases} \min \sum_{i \in \{1, \ldots, n\}} f(1 - x_i) \\ A \cdot \mathbf{x} \leq \mathbf{1} \\ x_i \geq 0, \ i \in \{1, \ldots, n\} \end{cases}$$

where, as previously, $A$ is the edge-vertex matrix of a graph $G$.

The approximation algorithm for CPm(\kappa) solves also the instances of IPm; consequently, $\hat{\nu}(f) \leq \rho \nu^*(f)$.

As for theorem 7, given a maximum independent set $S^*$ of $G$, $\mathcal{F}(S^*)$ is feasible for IPm and we have, $\forall f \in \mathcal{F}$,

$$\hat{\nu}(f) \leq \rho \nu^*(f) \leq \rho(n - \alpha(G))f(1). \quad (17)$$

On the other hand, let us consider the solution value $\hat{\nu}(f)$ (given by $A$) for IPm, once $G$, and consequently $A$, is given. We define $\hat{S}$ as in the proof of theorem 7 and we take it as an approximate solution for the maximum independent set problem on $G$. Since function $x \mapsto f(1 - x)$ is decreasing in $[0, 1]$, we have, $\forall f \in \mathcal{F}$, $\hat{\nu}(f) = \sum_{i \in \hat{S}} f(1 - \hat{x}_i(f)) + \sum_{i \notin \hat{S}} f(1 - \hat{x}_i(f)) \geq |\hat{S}|f(0) + (n - |\hat{S}|)f(1/2)$, or

$$|\hat{S}| \geq n \frac{\hat{\nu}(f)}{f(1/2)}. \quad (18)$$

Let us suppose now that in $G$, $n/\kappa \leq \alpha \leq n/2$. From expressions (17) and (18) we get, $\forall f \in \mathcal{F}$, $|\hat{S}|/\alpha \geq 2[1 - [\rho[1 - (1/\kappa)]f(1)/f(1/2)]]$ and since we have supposed that $S_\kappa$ is not polynomially constant-approximable we have, $\forall f \in \mathcal{F}$, $1 - [\rho[1 - (1/\kappa)]f(1)/f(1/2)] \leq 0$, or

$$\rho \geq [f(1/2)/f(1)][1/(1 - (1/\kappa))].$$

Since $\mathcal{F}$ satisfies the property described by expression (16), we easily get the following lower bound for $\rho$: $\rho \geq \kappa/[\kappa/(\kappa - 1)]\sup_{f \in \mathcal{F}} \{f(1/2)/f(1)\}$. \[ \]

From theorem 8, we can deduce the following negative result.

Corollary 4. The problem of minimizing a concave function in a polytope does not admit a polynomial time approximation algorithm guaranteeing an approximation ratio less than $\kappa/(\kappa - 1)$, unless $P = NP$.
Algorithm 3. The greedy S algorithm.

Part III

On the approximation ratio of the greedy algorithm of the maximum independent set.
always be a vertex of degree $\leq \Delta - 1$ to be selected to enter in $S'$. In fact, if before the deletion of all vertices from $V$ such a vertex does not exist, this implies that there is no vertex $v_i$ of $V$ having at least one common neighbour with a vertex already in $S'$, because if such a $v_i$ exists, then its degree is at most $\Delta - 1$. But, in this case, there is a set $V_2 \subseteq V$ (the set of the removed vertices during some steps of the algorithm) and a set $V_2 = V \setminus V_1$ such that there is no edges linking vertices of $V_1$ to vertices of $V_2$, contradicting so the hypothesis on the connectivity of $G$. Consequently, the cardinality $\alpha'(G)$ of the solution $S'$ satisfies $\alpha'(G) \geq \frac{[n - (\delta + 1)]}{\Delta} + 1$. Since $\delta \leq \Delta$, the above expression results to

$$\alpha'(G) \geq \frac{n - (\delta + 1)}{\Delta} + 1 \geq \frac{n - 1}{\Delta}. \quad (19)$$

From expression (2), we get:

$$\alpha(G) = n - m - (f + g) = m + e - (f + g) \quad (20)$$

where $e$ is the number of the exposed vertices (of $G$) with respect to a maximum matching of $G$. On the other hand, let us suppose that $f = n/x$; so, given that $G$ admits a perfect matching $M$, we have the following for the terms of the third part of expression (20):

$$m \leq \frac{n}{2},$$

$$e - g \leq 1,$$

$$f = \frac{n}{x}. \quad (21)$$

From expressions (19), (20) and (21), we obtain

$$\frac{\alpha'(G)}{\alpha(G)} \geq \frac{n - 1}{\Delta} + \frac{1}{\frac{n}{x}}. \quad (22)$$

The function on the righthand side of expression (22) is decreasing in $x$; on the other hand, by proposition 2, we can, without loss of generality, suppose that $f = n/x \geq 2$, or $x \leq n/2$; so, after some easy algebra, we get $\alpha'(G)/\alpha(G) \geq (2/\Delta) + [2/\Delta(n - 2)]$.

References


